A.P. STEPHENSON & D. J. STEPHENSON

Συστὴματα αρχειοθὲτησης και βὰσεις δεδομὲνων στον Amstrad

Για τους τύπους CPC - 464, 664, 6128



ΚΛΕΙΔΑΡΙΘΜΟΣ

A.P. STEPHENSON & D. J. STEPHENSON

Συστὴματα αρχειοθὲτησης και βὰσεις δεδομὲνων στον Amstrad

Για τους τύπους CPC - 464, 664, 6128

ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ
ΣΤΟΥΡΝΑΡΑ 27 Β
ΑΘΗΝΑ 106 82
ΤΗΛ: 3632044

Collins Professional and Technical Books
William Collins Sons & Co. Ltd
8 Grafton Street, London W1X 3LA

First published in Great Britain by
Collins Professional and Technical Books 1985

British Library Cataloguing in Publication Data
Stephenson, A. P.
Filing systems and databases for the
Amstrad CPC464.
1. Amstrad CPC464 (Computer) 2. Data base
management
I. Title II. Stephenson, D. J.
001.64'42 QA76.8.A4

Αποκλειστικότητα για την Ελληνική Γλώσσα:
ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ
Στουρνάρα 27 Β
106 82 Αθήνα, τηλ. 3632044

Περιεχόμενα

	Πρόλογος	5
1.	Εισαγωγή	7
2.	Συνθετικά Στοιχεία ενός Συστήματος διαχείρισης αρχείων..	22
3.	Απλά προγράμματα διαχείρισης Αρχείων και Δομικά Στοιχεία	47
4.	Ένα ολοκληρωμένο Σύστημα διαχείρισης Σειριακού Αρχείου Βασισμένο στη RAM	81
5.	Έρευνα και Ταξινόμηση	109
6.	Έλεγχος Γνώσεων	161
	Παράρτημα A: Γλωσσάρι	191
	Παράρτημα B: Κωδικοί ASCII	193
	Παράρτημα C: Απαντήσεις στις ασκήσεις	195

ΠΡΟΛΟΓΟΣ

Το βιβλίο αυτό αναφέρεται εξ ολοκλήρου στην αποθήκευση το χειρισμό και την ανάκτηση δεδομένων χρησιμοποιώντας το κασετόφωνο που είναι ενσωματωμένο στον Amstrad CPC464 ή το σύστημα δίσκων που υπάρχει στους CPC464 και 6128. Καθώς υποδηλώνει ο τίτλος, η έμφαση δίνεται στην έκδοση της BASIC για τον Amstrad και τον κώδικα μηχανής του Z80. Παρόλ' αυτά, τα περιεχόμενα μπορεί επίσης να ενδιαφέρουν και κατόχους άλλων υπολογιστών. Ελπίζεται ότι το βιβλίο θα ενδιαφέρει τόσο τον αρχάριο, όσο και τον έμπειρο χρήστη, υποτίθεται όμως ότι ο αναγνώστης θα έχει ήδη μερικές γνώσεις προγραμματισμού.

Αν και δίνονται ολόκληρες οι λίστες των κύριων προγραμμάτων, αυτά βασίζονται σε μια συλλογή από υπορουτίνες που είναι σχεδιασμένες να λειτουργούν σαν δομικά στοιχεία. Ελπίζεται ότι θα χρησιμοποιηθούν για να δημιουργήσετε άλλα προγράμματα με τροποποιήσεις και ιδιαίτερα γνωρίσματα για την αντιμετώπιση των δικών σας συγκεκριμένων αναγκών.

Το πρώτο κύριο πρόγραμμα είναι ένα σύστημα αρχειοθέτησης γενικής χρήσης, που προσφέρει μια μεγάλη σειρά βοηθητικών επεξεργασιών, ανάμεσά τους και την δυνατότητα δημιουργίας ενός υποαρχείου από ένα κύριο αρχείο (π.χ. την εξαγωγή από ένα κύριο αρχείο όλων των εγγραφών που έχουν κάποιες κοινές ιδιότητες, και τη συλλογή τους σε ένα ξεχωριστό υποαρχείο). Το δεύτερο πρόγραμμα που αποτελείται από δύο μέρη, δείχνει πως μπορεί να χρησιμοποιηθεί ο υπολογιστής για να κάνει τεστ πολλαπλής επιλογής (multiple choice) είτε σαν ελέγχους προόδου στα σχολεία είτε με τη μορφή κουίζ για το σπίτι. Έχουν συμπεριληφθεί και έλεγχοι ασφάλειας για να υποκατασταθεί μια τυπική σχολική ατμόσφαιρα. Εκτός από την εκπαιδευτική αξία, τα προγράμματα δείχνουν ότι τα αρχεία μπορούν να χρησιμοποιηθούν για την αποθήκευση ερωτήσεων και απαντήσεων καθώς και «εγγραφών» («records»).

Οι τεχνικές ταξινόμησης και αναζήτησης της BASIC περιγράφονται με λεπτομέρεια. Εν τούτοις, δίνονται και εναλλακτικές γρήγορες ρουτίνες σε κώδικα μηχανής μαζί με πλήρεις οδηγίες για τη σύνδεσή τους με τα κύρια

προγράμματα. Επειδή πολλοί χρήστες πιθανόν να μη διαθέτουν τις ευκολίες ενός assembler, δίνεται και ο αντικειμενικός κώδικας (object code) με τη μορφή «hex» (δεκαεξαδικών) προγραμμάτων φόρτωσης γραμμένων σε BA-SIC.

Μερικοί όροι αποκτούν σημασία με διαφορετική απόχρωση όταν εφαρμόζονται σε μικροϋπολογιστές. Για παράδειγμα οι βάσεις δεδομένων (data bases) και τα συστήματα αρχειοθέτησης (filing systems) είναι σαφώς διακριμένα στη βιβλιογραφία που απευθύνεται στον χρήστη των μεγάλων συστημάτων (mainframes) και των μίνι - υπολογιστών (minicomputers), αλλά, όσον αφορά τους οικιακούς μικροϋπολογιστές, οι περισσότεροι συγγραφείς αναφέρονται σ' αυτά σα να ήταν ένα και το αυτό. Πάλι, οι όροι που χρησιμοποιούνται για να διακρίνουν τα διάφορα συστήματα αρχειοθέτησης δεν χρησιμοποιούνται πάντα με συνέπεια αλλά προσπαθήσαμε να αποφύγουμε να υπεισέλθουμε σε υπερεξειδικευμένους ορισμούς.

A. P. Stephenson &
D. J. Stephenson

Κεφάλαιο 1

Εισαγωγή

Μέθοδοι Αρχειοθέτησης στο Σπίτι.

Οι περισσότεροι άνθρωποι στη σύγχρονη κοινωνία αισθάνονται την ανάγκη να αποθηκεύσουν πληροφορίες. Αν, προς στιγμήν, ορίσουμε σαν «πληροφορία», ό,τι μεταφέρει κάποιες έννοιες, και ένα «αρχείο» σαν μια «συλλογή σχετικών (μεταξύ τους) πληροφοριών», τότε έπεται ότι όλοι κρατάμε αρχεία με τη μία ή την άλλη μορφή, ακόμη κι αν είναι μόνο λογαριασμοί ηλεκτρικού ή νερού. Αξίζει να εξετάσουμε μερικές από τις μεθόδους αρχειοθέτησης που χρησιμοποιούνται στο σπίτι για να δούμε αν κάποια απ' αυτές θα μας βοηθήσει να σχεδιάσουμε συστήματα αρχειοθέτησης για υπολογιστή. Η επόμενη ταξινόμηση οπωσδήποτε δεν εξαντλεί το θέμα. Στην πραγματικότητα δεν αντιπροσωπεύει παρά μερικές απ' τις τεχνικές που χρησιμοποιούσαν οι συγγραφείς του βιβλίου αυτού πριν καταφύγουν στους υπολογιστές.

Το μοναδικό χαρτόκουτο

Σε πολλά σπίτια, το σύστημα οικιακής αρχειοθέτησης αποτελείται από ένα κουτί μόνο όπου σωρεύεται κάθε κομμάτι χαρτί που θεωρείται ότι έχει κάποια σημασία για το μέλλον. Έχει το υπέρτατο πλεονέκτημα ότι ελευθερώνει το μυαλό από την αγωνία για την λήψη αποφάσεων. Υπάρχει μόνο ένα κουτί, έτσι υπάρχει και ένα μόνο αρχείο. Τίποτα δεν θα ήταν απλούστερο.

Γυάλες γλυκού με επιγραφές

Αυτοί που περιφανεύονται ότι είναι μεθοδικοί έχουν την τάση να περιφρονούν το σύστημα του μοναδικού κουτιού με το επιχείρημα ότι στερείται ακριβείας. Προτιμούν να χρησιμοποιούν μια συστοιχία από γυάλες γλυκού, με μια καθαρή επιγραφή στην καθεμιά που αντιστοιχεί στη γενική φύση των περιεχομένων. Πληρωμένοι και απλήρωτοι λογαριασμοί μπορεί να ρίχνο-

νται σε μια γυάλα, οι μαγειρικές συνταγές σε άλλη και ίσως πιστοποιητικά γέννησης και γάμου, ή άλλα χαρτιά παρόμοιου τύπου, σ' ένα ελαφρώς καλύτερο είδος γυάλας. Εν τούτοις, η ταξινόμηση προϋποθέτει αποφάσεις. Από τη στιγμή που θα αρχίσεις μια μεθοδική προσέγγιση είναι αναπόφευκτο ο αριθμός από γυάλες να αρχίσει να αυξάνεται είτε γιατί (α) μερικές γυάλες γεμίζουν, ή (β) μερικά χαρτιά τυχαίνει να περιέχουν πληροφορίες που δεν ταιριάζουν με καμμιά από τις ήδη δημιουργημένες κατηγορίες, ή (γ) η αρχική απλή επιθυμία να εισάγεις μεθοδικότητα θα μπορούσε να μεταβληθεί σε φοβερή μονομανία. Αν συμβεί αυτό ο αριθμός των γυαλών μπορεί να αρχίσει να αυξάνεται απεριόριστα.

Ευτυχώς, η καταστροφή που προβλέπεται στην περίπτωση (γ) είναι σπάνια γιατί οι περισσότεροι γρήγορα ανακαλύπτουμε την αξία της επιγραφής για την κατηγορία *διάφορα*. Αυτή βολεύει για την καταχώρηση περίεργων χαρτιών που ανήκουν σε μια όχι ειδική κατηγορία. Η γυάλα με τα «διάφορα» είναι χρήσιμη σαν προσωρινή (;) θέση, με εκκρεμούσα μια τελική απόφαση.

Ο φάκελος εγγράφων γραφείου

Υπάρχουν ορισμένοι που περιφρονούν τόσο το κουτί όσο και τη γυάλα γλυκού. Προτιμούν να χρησιμοποιούν τον παραδοσιακό εξοπλισμό αρχειοθέτησης γραφείου - δηλαδή, άκαμπτους φακέλους με κομψές επιγραφές με την επικεφαλίδα του αρχείου και το σχετικό αριθμό, που περικλείουν τα αντίστοιχα χαρτιά. *Η διατήρηση της σειράς* εξασφαλίζεται με ένα μεταλλικό έλασμα στερεωμένο στο φάκελο, που συγκρατεί τα χαρτιά. Εν τούτοις, υπάρχει ελάχιστη διαφορά αρχών μεταξύ του φακέλου εγγράφων γραφείου και της γυάλας γλυκού αν και ο φάκελος είναι βολικός και προβάλλει μια πιο καλή εικόνα για τον χρήστη.

Το ιδανικό σύστημα αρχειοθέτησης

Δεν είναι εύκολο να ορίσουμε ένα ιδανικό σύστημα αρχειοθέτησης, αν και όλοι θα μπορούσαμε να συμφωνήσουμε με τις παρακάτω δοκιμαστικές προτάσεις:

(α) Το ιδανικό σύστημα αρχειοθέτησης πρέπει να έχει άπειρη χωρητικότητα αποθήκευσης.
(β) Πρέπει να είναι δυνατή η ανάκτηση κάθε ιδιαίτερου στοιχείου, ή στοιχείων, πληροφορίας από το αρχείο στιγμιαία και με ελάχιστη προσπάθεια.

Τα ιδανικά είναι χρήσιμα γιατί προκαλούν προσπάθειες για να τα πλησιάσουμε. Ας αρχίσουμε με την επιθυμία για «άπειρη χωρητικότητα αποθήκευσης». Αφήνοντας για μια στιγμή κατά μέρος το ότι είναι αδύνατο να πραγματοποιηθούν ποτέ, αξίζει να αναρωτηθούμε τι θα κάναμε την άπειρη χωρητικότητα αποθήκευσης, ακόμα κι αν την είχαμε. Σύμφωνα με την τρέχουσα άποψη «η πληροφορία είναι δύναμη». Το ποσό της διαθέσιμης πλη-

ροφορίας είναι τρομακτικό και φαίνεται να αυξάνεται σχεδόν εκθετικά με το χρόνο. Μέχρι το τέλος του Μεσαίωνα, ήταν δυνατό σε ένα άτομο με καλή μόρφωση και εξαιρετική μνήμη να γνωρίζει σχεδόν τα πάντα, όσα ήταν γνωστά. Τώρα, μια εικοσιτετράτομη εγκυκλοπαίδεια δεν είναι παρά ένα άτεχνο ευρετήριο για τη γνώση του κόσμου. Εν τούτοις, άλλο πράγμα είναι να έχεις όλη αυτή τη γνώση προσιτή σε τυπωμένη μορφή, και άλλο είναι το να βρεις ένα συγκεκριμένο στοιχείο της πληροφορίας. Αυτό μας οδηγεί στη δεύτερη επιθυμητή ιδιότητα που αναφέρθηκε παραπάνω - τη δυνατότητα να ανακτήσουμε μια συγκεκριμένη πληροφορία στιγμιαία και με την ελάχιστη προσπάθεια. Είναι σχεδόν προφανές ότι η δυσκολία ανεύρεσης μιας συγκεκριμένης πληροφορίας αυξάνεται αναλογικά με το σύνολο των αποθηκευμένων πληροφοριών. Όσο μεγαλύτερη είναι η βιβλιοθήκη, τόσο πιο δύσκολο είναι να βρεις το βιβλίο που θέλεις. Όσο ογκωδέστερο είναι το αρχείο, τόσο δυσκολότερο είναι να βρεις το χαρτί που θέλεις. Όσο μακρύτερη είναι η κασετοταινία τόσο περισσότερο χρόνο σου παίρνει να βρεις το bit που θέλεις.

Ο υπολογιστής σαν σύστημα αρχειοθέτησης

Ο πρώτος ψηφιακός υπολογιστής με αποθηκευμένο πρόγραμμα κατασκευάστηκε στο Πανεπιστήμιο του Καίμπριτζ το 1949 και λεγόταν EDSAC. Περιείχε 132.000 λυχνίες! Ο πρώτος υπολογιστής που παραδόθηκε στο εμπόριο ήταν ο UNIVAC το 1951. Αρχικά, οι κομπιούτερς έκαναν την εμφάνισή τους με βραδύ ρυθμό και απασχολούνταν κυρίως από μηχανικούς του δημοσίου και επιστήμονες για την επίλυση μαθηματικών προβλημάτων ή για την εκτέλεση κουραστικών αριθμητικών υπολογισμών σχετικών με βαλλιστικές τροχιές. Πέρασε μια δεκαετία πριν οι υπολογιστές αρχίσουν να χρησιμοποιούνται σε πιο γενικές εφαρμογές. Διαπιστώθηκε, μάλλον καθυστερημένα, ότι το πλήρες δυναμικό του υπολογιστή βρισκόταν στην ικανότητα του να αποθηκεύει και να ανακτά πληροφορίες. Με άλλα λόγια, η ικανότητα να επεξεργάζεται πληροφορίες ήταν εξίσου, αν όχι περισσότερο, σημαντική από την ικανότητα να κάνει υπολογισμούς. Σαν αποτελέσματα, εμφανίστηκε ένας νεολογισμός, *η επεξεργασία δεδομένων (data processing)*, για να τη διακρίνει από τους «συνηθισμένους» υπολογισμούς. Λόγω αυτής της αλλαγής που δίνεται στην έμφαση, *η χωρητικότητα αποθήκευσης* μιας μηχανής αποκτούσε μεγαλύτερη σημασία από την υπολογιστική της ικανότητα.

Η τεχνολογία της εσωτερικής μνήμης ανάγνωσης/γραφής (read/write memory), αυτής που τώρα ονομάζουμε *RAM*, εστιάστηκε γύρω από μία μήτρα μαγνητικών πυρήνων. Κάθε bit αποθηκευόταν σε ένα μικρό δακτύλιο από φερρίτη μέσα απ' τον οποίο περνούσαν τρία πάρα πολύ λεπτά σύρματα μεταφέροντας ρεύματα που καθόριζαν τον προσανατολισμό των μαγνητών είτε Βόρεια είτε Νότια (αντιπροσωπεύοντας τα δυαδικά ψηφία 1 και 0). Επειδή η μνήμη με μαγνητικούς πυρήνες απαιτούσε ένταση εργασίας για την κατασκευή της, το κόστος ήταν σχετικά τεράστιο παρά την απασχόληση εργα-

τικής δύναμης από τον Τρίτο Κόσμο. Εν τούτοις, είχε ένα πλεονέκτημα απέναντι στη σημερινή RAM ημιαγωγών. Ήταν μια μόνιμη μνήμη ανάγνωσης / γραφής. Μ' άλλα λόγια, διατηρούσε την πληροφορία ακόμη κι όταν σταματούσε η παροχή ρεύματος στον υπολογιστή επειδή τα bits ήταν αποθηκευμένα σαν σταγόνες μόνιμου μαγνητισμού. Η μονιμότητα αυτή θεωρούνταν σαν ένα είδος ιερών αγελάδων πριν την εμφάνιση, στα μέσα της δεκαετίας του '60, της RAM από ημιαγωγούς. Εν τούτοις, η χαμηλή τιμή και η χωρητικότητα της RAM ήταν αρκετή για να υπερκεράσει την αρχική αντίθεση και ο κόσμος των υπολογιστών βαθμιαία δέχτηκε την ιδέα μιας μη - μόνιμης εσωτερικής μνήμης.

Ο ερχομός της RAM είχε σαν αποτέλεσμα μια λεπτή αλλαγή στην ορολογία. Πριν την εποχή της RAM, η λέξη «μνήμη» σπάνια χρησιμοποιούνταν από τους ασχολούμενους στους υπολογιστές γιατί δεν τους άρεσε ο ανθρωπόμορφος συσχετισμός. Προτιμούσαν το γενικό όρο *αποθήκη (store)* αλλά έκαναν τη διάκριση ανάμεσα στην εσωτερική ή «κεντρική» αποθήκη, και την εξωτερική αποθήκευση στον περιφερειακό εξοπλισμό. Αυτή αναφερόταν σαν *μνήμη υποστήριξης (backing store).*

Μνήμη και αποθήκευση

Σήμερα, όταν αναφερόμαστε στη «μνήμη» θεωρείται σιωπηρά ότι εννοούμε την *RAM των ημιαγωγών*. Έτσι, όταν μιλάμε για μια συσκευή των 64K εννοούμε ότι είναι εγκατεστημένα 64 kilo-bytes. Η συμπληρωματική RAM του Amstrad αποτελείται από ένα τμήμα με 8 τσιπς καθένα απ' τα οποία έχει δυνατότητα αποθήκευσης 64K bits.

Τα περιφερειακά εξαρτήματα όπως η κασετοταινία, οι δίσκοι, ή οι σκληροί δίσκοι τύπου Winchester είναι «περιφερειακές μνήμες». Αρκετά είδη προγραμμάτων δεν κάνουν καθόλου χρήση της περιφερειακής μνήμης στη διάρκεια ενός «τρεξίματος» (RUN). Σ' αυτά τα προγράμματα η περιφερειακή μνήμη χρησιμοποιείται μόνο για να «αποθηκεύσει» (SAVE) το πρόγραμμα, πριν διακοπεί η παροχή ρεύματος. Απ' την άλλη μεριά προγράμματα που ασχολούνται με την επεξεργασία μεγάλων ποσοτήτων δεδομένων (όπως αυτά που χρειάζονται για τη διευθέτηση αρχείων) θα έχουν μεγάλες απαιτήσεις για την περιφερειακή μνήμη στη διάρκεια της χρήσης τους.

Όσον αφορά την περιφερειακή μνήμη, η πλειονότητα των κατόχων του Amstrad θα είναι ικανοποιημένη με το κασετόφωνο. Εν τούτοις, καθώς τα οικονομικά τους το επιτρέπουν ή καθώς αυξάνεται το ενδιαφέρον τους, ένα ποσοστό χρηστών μπορεί τελικά να επιλέξει τους δίσκους. Επειδή η περιφερειακή μνήμη έχει μεγάλη σημασία σε συστήματα αρχειοθέτησης, αξίζει να μελετήσουμε τις ιδιότητες των κασετών και των δίσκων.

Αποθήκευση σε κασετοταινία

Η κασετοταινία υπήρξε πραγματικό δώρο για τους χρήστες οικιακών υ-

πολογιστών. Αν γυρίσουμε πίσω στις πρώτες μέρες των οικιακών υπολογιστών, μερικοί τολμηροί τεχνικοί που είχαν σχέση με ένα Αμερικάνικο περιοδικό ονόματι *BYTE* αψήφησαν τα χλευαστικά σχόλια του κατεστημένου των υπολογιστών και κατόρθωσαν να συνδέσουν ένα κοινό κασετόφωνο με ένα μικρό υπολογιστή για την αποθήκευση προγραμμάτων. Το σύστημα αυτό έγινε γνωστό σαν «Πρότυπο του Κάνσας Σίτυ» (γιατί το *BYTE* εκδιδόταν σ' αυτή την πόλη) και, αφού υπέστη τροποποιήσεις και βελτιώσεις, έχει υιοθετηθεί ευρύτατα από τότε. Το βελτιωμένο σύστημα έγινε γνωστό σαν CUTS (Computer User's Tape Standard). Αν και το μεγαλύτερο μερίδιο της αναγνώρισης για τη διάδοση των οικιακών υπολογιστών ανήκει στον μικροεπεξεργαστή, είναι αμφίβολο αν θα είχαν γίνει τόσο δημοφιλείς χωρίς το ταπεινό κασετόφωνο. Η μονάδα κασέτας δεν είναι μόνο αρκετά φτηνή ώστε να είναι προσιτή στον περισσότερο κόσμο αλλά και το μέσο αποθήκευσης που χρησιμοποιεί (κενές κασέτες) είναι επίσης πολύ φτηνό.

Χωρητικότητα αποθήκευσης

Οι κασετοταινίες είναι διαθέσιμες σε διάφορα μήκη από C-30 μέχρι C-120 αλλά δεν συνίσταται η χρήση ταινιών μακρύτερων από C-30. Κασέτες ειδικά σχεδιασμένες για αποθήκευση δεδομένων των υπολογιστών έχουν συνήθως μήκος C-12. Όσο μακρύτερη είναι η ταινία, τόσο μεγαλύτερη η πιθανότητα να μαζευτεί ή να κοπεί.

Η ποσότητα των αποθηκευμένων δεδομένων μετριέται καλύτερα σε χαρακτήρες ASCII για μια δεδομένη μια δεδομένη ταχύτητα baud (baud rate) και μέχρις ενός σημείου, με το χρησιμοποιημένο πρωτόκολο μεταφοράς δεδομένων. Για να πάρουμε μια ιδέα της χωρητικότητας αποθήκευσης της κασέτας, ας υποθέσουμε ότι ένας χαρακτήρας ASCII απαιτεί 10 bits πληροφορίας (8 για τον κωδικό και 2 για σήμα αρχής/τέλους). Η ταχύτητα μεταφοράς δεδομένων (baud rate) είναι ουσιαστικά ο αριθμός bits πληροφορίας ανά δευτερόλεπτο (Για την ακρίβεια, δεν θα 'πρεπε να μιλάμε για «ταχύτητα baud» γιατί, όπως ο ναυτικός κόμβος, το baud είναι ήδη Ταχύτητα εκφρασμένη σε bits ανά δευτερόλεπτο).

Από τα παραπάνω, έπεται ότι:

Αποθηκευμένοι χαρακτήρες ανά δευτερόλεπτο = baud rate/10

Ο συνολικός αριθμός χαρακτήρων (Ν) που μπορεί να αποθηκευθεί επομένως σε μια ταινία μήκους (C) λεπτών είναι:

N = C x 60 x ταχύτητα baud/10

Παράδειγμα

Μια ταινία C-12 με ταχύτητα, ας πούμε, 2000 baud μπορεί να αποθηκεύσει 12x60x2000/10=144000 χαρακτήρες. Αυτό ισοδυναμεί με κάτι παραπάνω από 140K μνήμης. Στην πράξη, πρέπει να διατεθεί χώρος στο τέλος των ομάδων των κωδικών (block codes) ο οποίος θα χρησιμοποιηθεί από το λειτουργικό σύστημα της κασέτας καθώς και για bits που μπορεί να χαθούν στην αρ-

χή και στο τέλος της ταινίας οπότε είναι καλύτερα να δεχτούμε μια συντηρητική προσέγγιση και να κόψουμε τον αριθμό στη μέση. Αν το κάνουμε αυτό, μπορούμε να χρησιμοποιούμε το χοντρικό κανόνα ότι μια ταινία C-12, με ταχύτητα 2000 baud, μπορεί να αποθηκεύσει περίπου 70K πραγματικών δεδομένων.

Προβλήματα Ταχύτητας

Θα ήταν ανόητο να αρνηθεί κανείς ότι η αποθήκευση δεδομένων σε κασέτα είναι αργή. Πράγματι λέγεται συχνά (ιδιαίτερα από πωλητές δίσκων) ότι ένα σύστημα αρχειοθέτησης βασισμένο σε κασέτες είναι τελείως άχρηστο. Δεν συμμεριζόμαστε πάντα αυτή την άποψη. Τα συστήματα αρχειοθέτησης με κασέτα μπορούν να σχεδιαστούν με υψηλές προδιαγραφές και σίγουρα δεν είναι άχρηστα. Βασικό μειονέκτημα για το χρήστη του οικιακού μικροϋπολογιστή ή του μικροεπιχειρηματία που χρησιμοποιεί κασέτα είναι ότι κάποια αρχεία χρειάζονται μερικά λεπτά για να φορτωθούν. Είναι πολύ απλό να προγραμματίσεις ένα δυνατό χαρακτηριστικό ήχο όταν το φόρτωμα έχει τελειώσει, ώστε να μην χρειάζεται να κάθεσαι ακίνητος, παρακολουθώντας την ταινία να περνάει τις κεφαλές ανάγνωσης.

Η αποθήκευση σε κασέτα είναι αργή λόγω της ταχύτητας μεταφοράς της ταινίας και της σειριακής φύσης της προσπέλασης. Αν δεν είσαι ιδιαίτερα εξοικειωμένος με τα πλήκτρα για γρήγορο ξετύλιγμα/τύλιγμα, δεν μπορείς να πας κατ' ευθείαν στο bit που θέλεις. Θα δούμε αργότερα τα συστήματα αρχειοθέτησης τυχαίας προσπέλασης (random access filing systems) έχουν σημαντικά πλεονεκτήματα ταχύτητας αλλά, δυστυχώς, δεν μπορούν να εφαρμοστούν σε αρχεία που χρησιμοποιούν κασέτες.

Η BASIC στον AMSTRAD

Οι περισσότερες λίστες προγραμμάτων σ' αυτό το βιβλίο είναι στη BASIC αλλά, επειδή δεν υπάρχει κάτι σαν «πρότυπη» BASIC, αξίζει να αφιερώσουμε λίγο χώρο στην έκδοση της γλώσσας που χρησιμοποιεί ο Amstrad. Είναι μια έκδοση ισχυρή και με φαντασία, με μόνο μειονέκτημα την παράλειψη των procedures, που σημαίνει ότι είμαστε ακόμα προσκολημένοι στις εντολές GOSUB. Οι οριζόμενες συναρτήσεις (DEF FN) είναι διαθέσιμες υπό τον όρο ότι περιορίζονται σε μια μόνη γραμμή. Η πλειοψηφία των γνωστών εντολών (keywords), που αποτελούν το βασικό πυρήνα της γλώσσας, διατηρούνται ακόμα, οπότε δε χρειάζεται να τις ξαναπεριγράψουμε εδώ. Εν τούτοις, υπάρχει ένας αριθμός χρήσιμων πρόσθετων, που σπάνια βρίσκονται σε άλλες δημοφιλείς εκδόσεις, που έχουν άμεση σχέση με το χειρισμό αρχείων και την παρουσίαση δεδομένων. Είναι πιθανό όσοι είναι νεοφερμένοι στον κόσμο της γλώσσας των υπολογιστών να αισθανθούν ότι μερικές από τις επεξηγήσεις που υπάρχουν στο εγχειρίδιο του υπολογιστή βασίζονται σε μια περίεργη γλώσσα, τη Μεταγλώσσα, που είναι περιεκτική και κομψή αλλά μάλλον απαγορευτική ως προς την παρουσίαση. Η παρακάτω ε-

πιλογή εντολών (keywords) χρειάζεται περισσότερη επεξήγηση λόγω της ιδιαίτερης της συσχέτισης με την αρχειοθέτηση και γιατί μπορεί να μην είναι ήδη οικείες σε χρήστες άλλων εκδόσεων της BASIC.

Αυτόματες κλήσεις υπορουτινών

Η εντολή (keyword) EVERY σας επιτρέπει να εντάξετε σ' ένα πρόγραμμα μία *αυτόματη* κλήση μιας υπορουτίνας κατά κανονικά διαστήματα. Η κλήση χρησιμοποιεί οποιονδήποτε απ' τους τέσσερις υπάρχοντες μετρητές χρονικού διαστήματος. Η σύνταξη είναι η εξής:

EVERY χρονικό διάστημα, μετρητής GOSUB αριθμός γραμμής.

Το διάστημα είναι σε ακέραιες μονάδες του 1/50 του δευτερολέπτου. Οι μετρητές χαρακτηρίζονται 0, 1, 2, 3. Για παράδειγμα,

EVERY 50, 3 GOSUB 5000

Αυτή θα χρησιμοποιήσει τον Μετρητή 3 για να «πηδάει» στην υπορουτίνα της γραμμής 5000 κάθε 50 χρονικές μονάδες (1 δευτερόλεπτο). Αν αυτή η γραμμή γραφτεί μια φορά πιθανώς κοντά στην αρχή του προγράμματος, μπορεί να χρησιμοποιηθεί, ας πούμε για να *διακόπτει* το πρόγραμμα σε κανονικά διαστήματα για να εμφανίζει ένα μήνυμα στο κάτω μέρος της οθόνης. Μιλώντας γενικώτερα ο προγραμματιστής της BASIC μπορεί να χρησιμοποιήσει το *εσωτερικό* σύστημα διακοπής, μια δυνατότητα που συνήθως παρέχει μόνο η γλώσσα μηχανής. Η εκλογή τεσσάρων μετρητών σημαίνει ότι μπορούν να χρησιμοποιηθούν στο ίδιο πρόγραμμα τέσσερις ανεξάρτητες χρονικά προσδιορισμένες διακοπές.

Αποενεργοποίηση μιας διακοπής

Αν έχει προγραμματισθεί, με τη χρήση του EVERY, μια κατά χρονικά διαστήματα διακοπή (timed interrupt) μπορεί να συμβεί μια σύγκρουση λειτουργιών. Αν ένα ορισμένο μέρος του προγράμματος δεν πρέπει να διακοπεί με interrupt υπό οποιεσδήποτε συνθήκες είναι δυνατό να *αναχαιτίσουμε* την διακοπή χρησιμοποιώντας την εντολή DI (Disable Interrupt = Αποενεργοποίηση Διακοπής). Η αναχαίτιση θα διαρκέσει μέχρις ότου χρησιμοποιηθεί η εντολή EI (Enable interrupt = Ενεργοποίηση Διακοπής). Για παράδειγμα:

```
500 DI
510 τμήμα του προγράμματος που δεν πρέπει να διακοπεί.
..............
..............
600 EI
```

Τέλος Αρχείου

Η EOF είναι μια ψευδο-μεταβλητή που αντιπροσωπεύει το End Of File (= Τέλος Αρχείου). Όσο διαρκεί η είσοδος στοιχείων παραμένει στο 0 μέχρι να επισημανθεί το τέλος του αρχείου, οπότε και αλλάζει σε -1. Για παράδειγμα:

100 IF EOF<O, THEN κ.λπ., κ.λπ.

Ελεύθερη μνήμη

Όταν δημιουργούμε ένα καινούργιο αρχείο, είναι σκόπιμο να έχουμε μια συνεχή ενημέρωση πάνω στην ποσότητα της μνήμης που είναι ελεύθερη. Ένα από τα μειονεκτήματα των περισσότερων interpreters της BASIC (η έκδοση για τον Amstrad δεν αποτελεί εξαίρεση) είναι ο τρόπος με τον οποίο αποθηκεύονται οι αλφαριθμητικές παραστάσεις (strings). Ακατάστατοι σωροί με άχρηστες πληροφορίες τείνουν να διαμοιραστούν σ' όλη τη RAM - υπολείμματα αλφαριθμητικών μεταβλητών που δεν συμπληρώθηκαν. Όταν δεν υπάρχει πια διαθέσιμος χώρος, το λειτουργικό σύστημα αναλαμβάνει μια διαδικασία γνωστή σαν *συλλογή αχρήστων (garbage collection)* για να ανακτήσει ελεύθερο χώρο. Αυτό μπορούμε να το προβλέψουμε χρησιμοποιώντας την ψευδο-μεταβλητή FRE.

Η FRE (« ») πραγματοποιεί τη συλλογή αχρήστων πριν καταμετρήσει τα bytes ελεύθερης μνήμης. Για παράδειγμα, η PRINT FRE (« ») μπορεί να χρησιμοποιηθεί σε οποιοδήποτε σημείο του προγράμματος. Αυτή η διαδικασία χρειάζεται κάποιο χρόνο (ίσως ορισμένα δευτερόλεπτα) που εξαρτάται από την ποσότητα των δεδομένων που πρέπει να συλλεχτούν. Η FRE (0) δίνει τα ελεύθερα bytes αλλά χωρίς συλλογή αχρήστων.

Περιορισμός της περιοχής της μνήμης που χρησιμοποιείται από τη BASIC

Η περιοχή της μνήμης που βρίσκεται πάνω απ' αυτή που χρησιμοποιεί η BASIC είναι, θεωρητικά, διαθέσιμη για άλλους σκοπούς — για παράδειγμα, για υπορουτίνες σε γλώσσα μηχανής. Η διεύθυνση στην κορυφή της περιοχής που τώρα καταλαμβάνεται από τη BASIC είναι πάντα διαθέσιμη στη δυναμική μεταβλητή HIMEM. Ο προγραμματιστής μπορεί να περιορίσει το χώρο της BASIC εκχωρώντας μια σταθερή διεύθυνση στην ψευδο-μεταβλητή MEMORY. Για παράδειγμα, η MEMORY &2000 θα καθορίσει το ανώτερο όριο διεύθυνσης για τη BASIC. Η μνήμη πάνω από αυτή, που κανονικά ανήκει στη BASIC, απελευθερώνεται τότε για τη γλώσσα μηχανής.

Εύρεση μιας αλφαριθμητικής μέσα σε μια άλλη αλφαριθμητική παράσταση

Η εντολή (keyword) INSTR είναι χρήσιμη για να καθορίσουμε αν μια

ορισμένη ομάδα χαρακτήρων παρουσιάζεται μέσα σε μια αλφαριθμητική μεταβλητή. Η σύνταξη, με χρήση αυθαίρετων μεταβλητών, είναι INSTR (A$,B$) όπου B$ είναι η υπο-αλφαριθμητική μεταβλητή (substring) για την οποία ψάχνουμε μέσα στην A$. Αν η B$ είναι πράγματι μέσα στην A$, η εντολή επιστρέφει στη θέση της αλφαριθμητικής μεταβλητής στην A$, εκεί όπου αρχίζει η B$. Αν η B$ δε βρεθεί καθόλου, τότε ο επιστρεφόμενος αριθμός είναι το μηδέν. Για παράδειγμα,

100 IF INSTR (A$,B$)= O THEN PRINT "Η Υποαλφαριθμητική μεταβλητή δεν υπάρχει".

Αυτό μπορεί να συνεισφέρει στο θέμα της «φιλικότητας προς το χρήστη» (userfriendliness) σε μια ρουτίνα που ερευνάει ένα αρχείο για μια δεδομένη εγγραφή. Ένας τελικός χρήστης μπορεί να θυμάται μερικούς χαρακτήρες ενός στοιχείου που ψάχνει αλλά όχι όλους, έτσι μια δυνατότητα αυτής της μορφής θα είναι χρήσιμη.

Μετατροπή κεφαλαίων και μικρών χαρακτήρων

Είναι δυνατό να εξασφαλίσουμε ότι όλοι οι χαρακτήρες σε μια αλφαριθμητική παράσταση είναι είτε κεφαλαίοι είτε μικροί. Οι δύο αντίστοιχες εντολές είναι UPPER$ και LOWER$. Για παράδειγμα

A$ = UPPER$ (B$) θα έχει σαν αποτέλεσμα η A$ να περιέχει ολόκληρη τη B$ με κεφαλαία.
A$ = LOWER$ (B$) θα έχει σαν αποτέλεσμα μια ισοδύναμη γραφή με μικρά.

Ένα χρήσιμο παράδειγμα είναι εκείνο όπου ο χειριστής πρέπει να εισάγει μίαν απάντηση του τύπου ναι/όχι σε κάποιο ερώτημα. Η εισαχθείσα απάντηση μπορεί να είναι σε κεφαλαία ή μικρά, οπότε θα χρειαζόταν μια μεγάλη γραμμή IF/THEN για να καλύψει και τις δύο περιπτώσεις. Η UPPER$ μπορεί να χρησιμοποιηθεί για να μετατρέψει τα δεδομένα εισόδου σε κεφαλαία καθιστώντας έτσι μη αναγκαία τη γραμμή IF/THEN. Μπορούμε να σκεφτούμε και πολλά άλλα παραδείγματα, όπως η ταξινόμηση (σορτάρισμα) αλφαριθμητικών παραστάσεων όταν το πρώτο γράμμα της καθεμιάς αλφαριθμητικής μπορεί να είναι είτε κεφαλαίο είτε μικρό.

Εναλλακτική λύση για το βρόχο FOR/NEXT

Ο τυπικός βρόχος FOR/NEXT υπήρξε ανέκαθεν η ραχοκοκκαλιά της γλώσσας BASIC για τον προγραμματισμό επαναλαμβανόμενων σειρών εντολών. Αν και η BASIC του Amstrad την περιλαμβάνει, φυσικά, παρέχεται μια εναλλακτική και συχνά προτιμητέα λύση που λέγεται βρόχος WHILE/WEND. Η αρχή του βρόχου ορίζεται από τη συνθήκη WHILE και το τέλος από την εντολή WEND (που σημαίνει «While END» = «τέλος του while»).

Το σώμα του βρόχου συνεχίζει να εκτελείται μέχρις ότου η συνθήκη, ή οι συνθήκες, ικανοποιηθούν. Αν και θεωρείται κακός προγραμματισμός να βγαίνει κανείς έξω από ένα βρόχο FOR/NEXT πριν το κανονικό του τέρμα, είναι επιτρεπτό να αποφύγουμε το WEND στους βρόχους WHILE/WEND.

Το κασετόφωνο του Amstrad

Η διαδικασία για τη χρήση της κασέτας δίνεται στις οδηγίες χρήσης που συνοδεύουν τη συσκευή αλλά οι πρόσθετες πληροφορίες που ακολουθούν μπορούν να σας βοηθήσουν να αποκομίσετε το μέγιστο όφελος από το κασετόφωνο.

Το κασετόφωνο για την πλειοψηφία των οικιακών υπολογιστών είναι μια ιδιαίτερη συσκευή που πρέπει να αγοραστεί ξεχωριστά. Η ενσωματωμένη μονάδα στον Amstrad CPC 464 έχει τα ακόλουθα πλεονεκτήματα, πέρα και πάνω από την προφανή οικονομία του επί πλέον κόστους:

(α) Δεν υπάρχουν ενοχλητικά καλώδια για συνδέσεις και, κατά συνέπεια, ούτε φασαρία με προβληματικές συνδέσεις σε πρίζες.

(β) Λόγω του ολοκληρωμένου σχεδιασμού, τα *επίπεδα του σήματος ανάμεσα στον υπολογιστή και το κασετόφωνο είναι βελτιστοποιημένα και σταθερά* (όχι χασομέρια με έλεγχους της έντασης και του τόνου).

(γ) Σαν άμεσο αποτέλεσμα των σταθερών επιπέδων σήματος, είναι ότι η ταχύτητα μεταφοράς δεδομένων μπορεί να είναι υψηλή - 2000 baud στην πράξη!

Υπάρχει δυνατότητα επιλογής ανάμεσα σε δύο ταχύτητες ταινίας, την κανονική ταχύτητα των 1000 baud ή την επιλέξιμη από το πρόγραμμα υψηλότερη ταχύτητα των 2000 baud. Η διαδικασία γι' αυτή την επιλογή, που γίνεται με την εντολή SPEED WRITE, είναι η εξής:

Για να ορίσετε ταχύτητα 1000 baud, γράψτε SPEED WRITE O
Για να ορίσετε ταχύτητα 2000 baud, γράψτε SPEED WRITE 1

Οι τιμές αυτές είναι ονομαστικές όπως είναι στην πραγματικότητα όλες οι τιμές ταχύτητας baud, γιατί υποθέτουν ότι τα μεταφερόμενα δεδομένα συνίστανται από κανονικές εναλλαγές των δυαδικών bits, 1 και O. Μ' άλλα λόγια, η αναφερόμενη ταχύτητα baud βασίζεται, πολύ λογικά, σε *στατιστικούς* μέσους όρους. Ένας άλλος παράγοντας που συχνά αμελείται είναι η παρουσία των δεδομένων τυποποίησης (formatting data), όπως είναι οι *χαρακτήρες που δείχνουν το τέλος του μπλοκ (end of block characters),* καθώς και τα *κενά μεταξύ των μπλοκ (inter - block pauses)* κατά τη μεταφορά των δεδομένων. Όταν ο Amstrad ξαναδιαβάζει ένα αρχείο από ταινία, χρησιμοποιεί την ταχύτητα baud που χρησιμοποιήθηκε όταν έγινε η εγγραφή του αρχείου. Οι εντολές LOAD «όνομα αρχείου» ή INPUT «όνομα αρχείου» θα χειριστούν τη διαδικασία ανάγνωσης, ανεξάρτητα από την ταχύτητα baud που χρησιμοποιήθηκε κατά την αποθήκευση του αρχείου.

Ταχύτητα αποθήκευσης πληροφοριών

Καλά είναι όλα όσα έχουν αναφερθεί για τις ταχύτητες baud αλλά αν είσαστε νεοφερμένοι στους υπολογιστές και ο Amstrad είναι η πρώτη σας συσκευή, θα εκτιμούσατε μια πιο ρεαλιστική μέτρηση της ταχύτητας της ταινίας. Πιθανόν να θέλετε να ξέρετε πόσο χρόνο θα χρειαστεί η αποθήκευση μιας συνηθισμένης σελίδας κειμένου A4. Για να απαντηθεί αυτό, πρέπει να αρχίσουμε από το χρόνο που απαιτείται για να αποθηκευτεί ένας χαρακτήρας. Ένας χαρακτήρας του κώδικα ASCII απαιτεί 8 bits, αλλά, όπως δηλώθηκε προηγούμενα, χρειάζεται τουλάχιστον άλλο ένα ζευγάρι bits για να σημάνει την αρχή και το τέλος κάθε χαρακτήρα, οπότε καταλήγουμε προσωρινά στη στρογγύλευση των 10 bits ανά χαρακτήρα. Μια ταχύτητα ταινίας 2000 baud (2000 bits ανά δευτερόλεπτο) σημαίνει ότι η ταχύτητα χαραχτήρων είναι 2000/10=200 χαρακτήρες ανά δευτερόλεπτο ή 12000 χαρακτήρες ανά λεπτό. Αν υποθέσουμε ότι η μέση λέξη στην Αγγλική γλώσσα περιέχει πέντε γράμματα, αυτό ισοδυναμεί με 12000/5= 2400 λέξεις ανά λεπτό. Μια συνηθισμένη σελίδα τυπωμένου κειμένου με διπλό διάστημα μεταξύ των γραμμών περιέχει περίπου 28 γραμμές με 13 λέξεις ανά γραμμή= 364 λέξεις ανά σελίδα. Με 2400 λέξεις ανά λεπτό, σημαίνει ότι η ταχύτητα μεταφοράς δεδομένων μέσω κασετοταινίας είναι εξήμιση περίπου σελίδες τυπωμένου κειμένου ανά λεπτό. Αυτό δεν φαίνεται πολύ γρήγορο, αλλά, πριν αρχίσουμε να γκρινιάζουμε πρέπει να κάνουμε τη διάκριση ανάμεσα στη συνηθισμένη σελίδα κανονικού κειμένου και τη συνηθισμένη εγγραφή για υπολογιστή. Μια «κανονική σελίδα» κειμένου όπως θα τη βρίσκατε σ' ένα μυθιστόρημα, περιέχει μεγάλο ποσοστό από *παραγεμίσματα.* Μπορεί να είναι ευχάριστο, μπορεί να είναι διασκεδαστικό και μπορεί ακόμη να είναι σημαντικό, γι' αυτούς που έχουν κλίση στη λογοτεχνία, αλλά, όσον αφορά την προσθήκη πληροφοριών πραγματικής αξίας καθώς προχωρούν οι σελίδες, δεν είναι παρά παραγεμίσματα. Αντίθετα, μια συνηθισμένη εγγραφή σε υπολογιστή είναι, ή θα 'πρεπε να είναι, σχεδόν παραγεμίσματα. Κάθε λέξη πρέπει να έχει το βάρος της και πρέπει να ελαχιστοποιηθούν τα τερτίπια και οι αστεϊσμοί του λόγου. Πρέπει να είναι καθαρή πληροφορία. Κάθε λέξη που θα μπορούσε να αφαιρεθεί χωρίς να καταστρέψει την αξία της πληροφορίας δεν θα έπρεπε να ήταν εκεί. Είναι εκπληκτικό πόσο πολλές πληροφορίες μπορούν να συμπυκνωθούν σε μια εγγραφή που περιέχει μόνο πενήντα λέξεις. Για παράδειγμα, σε ένα καλά-σχεδιασμένο τρόπο εγγραφής, πενήντα λέξεις είναι φυσιολογικά υπεραρκετές για μια ατομική εγγραφή που περιέχει το όνομα, πλήρη διεύθυνση, ταχυδρομικό κώδικα, αριθμό τηλεφώνου, φύλο, επάγγελμα και πιστωτικό υπόλοιπο. Ίσως να υπάρχουν ακόμη αρκετές ελεύθερες λέξεις για την αποθήκευση ορισμένων ιδιαίτερων χαρακτηριστικών. Παίρνοντας σαν παράδειγμα τις εγγραφές των πενήντα λέξεων, επιτυγχάνουμε μια ταχύτητα μεταφοράς της ταινίας που πλησιάζει τις 2400/50 = 48 εγγραφές ανά λεπτό, που είναι λογική και μας εξοπλίζει να αντιμετωπίσουμε εκείνους με τα παχειά πορτοφόλια που συνήθως χλευάζουν τα αρχεία σε κασετοταινίες.

Επιλογή ταχύτητας baud

Γενικά, όσο υψηλότερη είναι η ταχύτητα baud τόσο μεγαλύτερη είναι πιθανότητα ενός λάθους κατά την ανάγνωση, έτσι άν τα δεδομένα που επιθυμείτε να αποθηκεύσετε είναι ιδιαίτερα πολύτιμα, θα ήταν πιθανώς καλύτερο να χρησιμοποιήσετε την χαμηλή ταχύτητα των 1000 baud. Εν τούτοις, θα 'πρεπε να σημειώσουμε ότι χρησιμοποιήσαμε την υψηλή ταχύτητα συνεχώς κατά τη συγγγραφή αυτού του βιβλίου και δεν συναντήσαμε ακόμη κανένα λάθος ανάγνωσης. Φαίνεται να είναι ένα εξαιρετικά αξιόπιστο σύστημα. Η αξιοπιστία της ταχύτητας των 2000 baud δεν είναι και τόσο καλή αν ένα πρόγραμμα εγγραφεί σε μία συσκευή και δουλέψει σε μια άλλη. Αυτό συμβαίνει γιατί μπορεί να υπάρχουν ελαφρές διαφορές στην ευθυγράμμιση ή στις ιδιότητες των μαγνητικών κεφαλών. Στην ίδια συσκευή, αυτό δεν είναι τόσο σημαντικό, γιατί είναι ίδιες τόσο κατά την εγγραφή όσο και κατά την ανάγνωση. Έτσι, αν έχετε την πρόθεση να διαβαστεί το πρόγραμμά σας σε συσκευές άλλων, θα ήταν ίσως ασφαλέστερο να εγγράφετε στα 1000 baud.

Κίνδυνοι της αρχής της ταινίας

Η πλειοψηφία των κασετών που διατίθενται στα καταστήματα έχουν πολλά εκατοστά κενής *αρχής ταινίας (leader tape)* πριν από την κανονική περιοχή την επενδυμένη με οξείδιο. Η αρχή της ταινίας δημιουργεί μια ισχυρή μηχανική σύνδεση με το καρούλι περιέλιξης και καθαρίζει την κεφαλή εγγραφής / ανάγνωσης, αλλά, φυσικά, δεν μπορεί να αποθηκεύσει πληροφορίες. Αν αρχίσετε τη διαδικασία εγγραφής πολύ νωρίς - δηλαδή, πριν περάσει η αρχή της ταινίας την κεφαλή εγγραφής - το πρώτο τμήμα των δεδομένων σας μπορεί να χαθεί και θα είναι αδύνατο να το επανακτήσετε όταν προσπαθήσετε αργότερα να ξαναφορτώσετε. Για να προλάβετε μια τέτοια καταστροφή, αξίζει να πατήσετε το πλήκτρο PLAY για δυό δευτερόλεπτα μετά από ένα πλήρες ξανατύλιγμα ώστε η αρχή της ταινίας να περάσει τις κεφαλές πριν αρχίσετε τη διαδικασία εγγραφής. Μια καλή μέθοδος είναι να προχωρήσετε την ταινία μέχρι ο ψηφιακός μετρητής του κασετόφωνου να φθάσει το 5. Μπορείτε να αγοράσετε κασέτες χωρίς την αρχή που περιγράψαμε παραπάνω αλλά δε βρίσκονται τόσο εύκολα. Οι κασέτες χωρίς «αρχή ταινίας» μπορεί να σας οδηγήσουν σε μια λανθασμένη αίσθηση ασφάλειας αλλά και πάλι θα σας συμβουλεύαμε να περιμένετε τουλάχιστο ένα δευτερόλεπτο για να επιτρέψετε να περάσουν μερικά εκατοστά από την αρχή της ταινίας. Αυτά τα πρώτα λίγα εκατοστά μπορεί να είναι κακής ποιότητας λόγω της σφιχτής περιέλιξης γύρω από το καρούλι. Οι λεγόμενες ταινίες «ψηφιακής ποιότητας» (digital quality) συχνά έχουν καρούλια μεγαλύτερης διαμέτρου, ώστε η περιέλιξη να είναι λιγότερο σφιχτή στα εσωτερικά στρώματα. Τελικά, αυτό θα λέγαμε σαν προειδοποίηση - σκεφτείτε προσεκτικά πριν αγοράσετε κασέτες σε τιμές ευκαιρίας. Η εξοικονόμηση ενός μικρού ποσού απ' την τιμή μιας κασέτας συχνά μπορεί να είναι λάθος οικονομία. Οι κασέτες καλής ποιότητας, ιδιαίτερα αυτές που είναι σχεδιασμένες σαν

κατάλληλες για ψηφιακά δεδομένα, είναι κατασκευασμένες με στενότερα ό-ρια ανοχής οπότε υπάρχουν πολύ λιγότερες δυνατότητες ύπαρξης *αλλοιωμένων* περιοχών πάνω στην επιφάνεια οξειδίου. Παρόμοιες αλλοιώσεις (ανωμαλίες στην επένδυση οξειδίου) είναι σχετικά ασήμαντες κατά την εγγραφή κανονικής μουσικής αλλά ένα bit ανεστραμμένο σε μια ταινία δεδομένων θα μπορούσε να οδηγήσει σε χάος.

Εφεδρικά αντίγραφα (Back-up copies)

Αν μια ταινία αποτύχει στο να φορτώσει ένα απ' τα προγράμματά σας το χειρότερο που μπορεί να συμβεί είναι εκνευρισμός και πολλή πρόσθετη δουλειά για να πληκτρολογήσετε ξανά τη λίστα του προγράμματος. Εν τούτοις, αν η αποτυχία τυχαίνει να αφορά μια ταινία δεδομένων που περιέχει σημαντικές επιχειρηματικές ή ιδιωτικές εγγραφές, οι συνέπειες είναι πολύ σοβαρότερες. Ένα αρχείο λίγων εκατοντάδων εγγραφών μπορεί να αντιπροσωπεύει πολλές ώρες προσεκτικής εργασίας για την εισαγωγή πληροφοριών, που μερικές μπορεί ποτέ να μην είναι πάλι διαθέσιμες. Αν και το κασετόφωνο του Amstrad φαίνεται πολύ αξιόπιστο, η απώλεια δεδομένων μπορεί μερικές φορές να συμβεί είτε λόγω κακής κασέτας είτε λόγω λαθεμένης πληκτρολόγησης από μέρους του χειριστή κατά την αρχική εγγραφή ή την εν συνεχεία ανάγνωση. Είναι πολύ εύκολο, ιδιαίτερα όταν αισθάνεται κανείς κουρασμένος, να πατήσει το PLAY αντί για τα PLAY και REC έτσι ώστε τίποτα να μη γίνει στην ταινία. Μπορεί να αγνοεί το λάθος μέχρι να προσπαθήσει αργότερα να ξαναφορτώσει την ταινία.

Για να αποφύγετε την οργή και την απογοήτευση, κρατάτε πάντα δύο ή ακόμη και τρία αντίγραφα στην ίδια ταινία και, για πρόσθετη ασφάλεια, έχετε επί πλέον αντίγραφα σε ξεχωριστή ταινία. Παίρνοντας δύο αντίγραφα, το ένα μετά το άλλο, στην ίδια κασέτα, ή στην αντίθετη πλευρά, σίγουρα θα μειώσετε τον κίνδυνο αλλά υπάρχει πάντα η πιθανοτητα να παραπεταχτεί, να πέσει σε μια σκληρή επιφάνεια ή να τοποθετηθεί στο σκέπασμα ενός ηχείου hi-fi. Οι μαγνήτες, ιδιαίτερα των ηχείων, έχουν φθοροποιό επίδραση στα δεδομένα της κασέτας και τα δύο τους θα 'πρεπε να κρατιούνται σε ασφαλή μεταξύ τους απόσταση. Υποθέτοντας ότι δε θα μπορούσε να συμβεί το ίδιο ατύχημα σε δύο διαφορετικές κασέτες, θα 'πρεπε να αποφασίσετε, παρά την πρόσθετη δαπάνη, να κρατήσετε ένα εφεδρικό αρχείο σε μια άλλη κασέτα. Δεν είναι καλή τακτική να στοιβάζετε διαφορετικά προγράμματα και περίεργα υπολείμματα δεδομένων σε μια κασέτα γιατί (α) μπορεί να χρειαστεί κάμποσος χρόνος για να εντοπίσετε το κομμάτι που θέλετε και (β) δημιουργεί προβλήματα με τις ετικέτες της κασέτας.

Αν δε σας πειράζει να χάνετε αχρησιμοποίητη ταινία, ο καλύτερος κανόνας που μπορείτε να ακολουθήσετε είναι να χρησιμοποιείτε *μια ξεχωριστή ταινία για κάθε πρόγραμμα ή αρχείο δεδομένων*. Όλες αυτές οι συμβουλές μπορεί να σας δημιουργήσουν την εντύπωση ότι η διατήρηση αρχείων του υπολογιστή σε ταινία μπορεί να είναι παράτολμο εγχείρημα. Όπως έχει ήδη δηλωθεί, το σύστημα ταινίας του Amstrad είναι άκρως αξιόπιστο αλλά αυτό

δεν αποτελεί δικαιολογία για να μη ληφθεί κάθε δυνατή προφύλαξη για να διατηρηθεί η ακεραιότητα των αρχείων. Ακόμη και σε ένα μεγάλο επαγγελματικό υπολογιστικό συγκρότημα, που χρησιμοποιεί υλικά υψηλού κόστους και άψογης αποδοτικότητας, η λήψη αντιγράφων των αρχείων μαζί με τη διατήρηση και την οργάνωση τους, επιβάλλεται σχεδόν με στρατιωτική πειθαρχία.

Ετικέτες

Η σημασία των επιγραφών αυξάνεται ανάλογα με τον αριθμό των εν χρήσει κασετών ή δισκετών. Αν έχετε μόνο δύο ή τρεις ταινίες, είναι ίσως απώλεια χρόνου το να γράφετε ετικέτες, γιατί συνεχώς θα αλλάζετε τα περιεχόμενα. Καθώς η συλλογή σας μεγαλώνει, ή αν χρησιμοποιείτε τη συσκευή για να κρατάτε σημαντικές εγγραφές, είναι σκόπιμο να υιοθετήσετε μια μεθοδική προσέγγιση στο θέμα των ετικετών. Μπορείτε να αγοράσετε ετικέτες σχεδιασμένες για κασέτες αλλά, κατά τη γνώμη μας, αυτές είναι λίγο ακριβές. Η καλύτερη λύση είναι να αγοράσετε ένα ρολό απλού αυτοκόλλητου για επιγραφές, και να τις κόβετε στα μεγέθη που τις θέλετε. Κολήστε τις επιγραφές στην ίδια την κασέτα, όχι στο πλαστικό κουτί, γιατί (α) η κασέτα θα μπορούσε να μπεί σε λάθος θήκη, και (β) άπαξ και κολήσετε μια ετικέτα στη θήκη, είναι πολύ δύσκολο να την αφαιρέσετε χωρίς ξύσιμο και λέρωμα. Αφ' ετέρου οι ετικέτες φαίνεται να ξεκολιούνται εύκολα και καθαρά από το σώμα της κασέτας. Αν αλλάξει το περιεχόμενο της κασέτας, είναι καλύτερο να αντικαταστήσετε όλη την επιγραφή παρά να κάνετε μεταβολές. Μια επιγραφή με πολλές μεταβολές είναι άσχημη και εμπνέει λιγότερη εμπιστοσύνη. 'Αλλη μέθοδος είναι απλώς να επιγράφετε τις κασέτες 1, 2, 3, κ.λπ. και να έχετε σε κατάλογο τα περιεχόμενα σε ένα βιβλίο Καταγραφής Ταινιών. Το πρόβλημα με τις καταγραφές είναι ότι ο αρχικός ενθουσιασμός για τη διατήρησή τους εξαφανίζεται συχνά. Πραγματικά δεν έχει σημασία ποιό σύστημα χρησιμοποιείτε, αρχεί να έχετε κάποιο και να τη τηρείτε.

Ομάδες δεδομένων (data blocks) σε κασέτα

'Ενα κασετόφωνο, επειδή είναι μηχανικό σύστημα, λειτουργεί σε μια χρονική κλίμακα που είναι χιλιάδες φορές αργότερη από τον υπολογιστή. Για να αντιμετωπίσει τη χρονική διαφορά, ο Amstrad δεσμεύει μια περιοχή μνήμης των 2K, που λέγεται *μπλοκ* και που χρησιμοποιείται σαν *buffer* ανάμεσα στην κασέτα και τον υπολογιστή. Οι μεταφορές δεδομένων ανάμεσα στον υπολογιστή και στην κασέτα εκτελούνται από και μέσα σ' αυτή την περιοχή του buffer. 'Οταν φορτώνεται ένα πρόγραμμα, το μήνυμα στην οθόνη δείχνει τον αριθμό καθενός μπλοκ των 2K.

Η Κατανομή της περιοχής του buffer της κασέτας

Ο Amstrad χρησιμοποιεί τη *δυναμική* κατανομή (dynamic allocation)

για αρχεία σε κασέτες. Αυτό σημαίνει ότι παραχωρείται χώρος στον buffer της κασέτας μόνο όταν χρειάζεται. Υπάρχουν προφανείς λόγοι για τους οποίους οι σχεδιαστές προτίμησαν τη δυναμική κατανομή από τη συμβατική σταθερή περιοχή του buffer. Πολλά προγράμματα μπορεί να μη χρησιμοποιούν αρχεία δεδομένων σε κασέτες, έτσι γιατί να χάνεται πολύτιμη RAM κρατώντας χώρο που μπορεί να μη χρησιμοποιηθεί; Κάποια στιγμή, μπορεί να συνδεθεί ένα σύστημα δίσκου στον CPC464 ή να έχετε τον CPC664 ή CPC6128 κάνοντας έτσι περιττή τη μονάδα κασέτας. Ο Amstrad μπορεί να υποστηρίξει δύο ανοιχτά αρχεία (open files) σε κασέτες. Για το κάθε ένα buffer εισόδου και εξόδου, χρειάζεται μια περιοχή buffer των 2K, που κάνει συνολικά 4K.

Η υψηλότερη διαθέσιμη θέση μνήμης, που υπάρχει στη BASIC αποθηκεύεται στη μεταβλητή HIMEM. Οι αλφαριθμητικές παραστάσεις αποθηκεύονται από τη HIMEM και προς τα κάτω και αναφέρονται σα «σωρός» (heap). Για να κρατηθούν οι buffer της κασέτας, η BASIC εκτελεί μια διαδικασία νοικοκυρέματος ώστε να εξασφαλίσει ότι ο σωρός είναι όσο το δυνατό μικρότερος και τότε τον μετακινεί στη μνήμη κατά 4K προς τα κάτω. Η HIMEM παίρνει ακολούθως αυτή τη χαμηλότερη τιμή. Όταν ο buffer της κασέτας έχει εκπληρώσει τον προορισμό του, η BASIC προσπαθεί να ανακτήσει τη μνήμη που καταλήφθηκε από τον buffer. Αυτό μπορεί να το κάνει μόνο αν η περιοχή του buffer είναι αμέσως πάνω από τη HIMEM. Μ' αυτή την προϋπόθεση, ο σωρός μετακινείται πίσω στην αρχική του θέση και η HIMEM ξαναπαίρνει την λανθάνουσα τιμή της.

Η δυναμική κατανομή των buffers της κασέτας δουλεύει θαυμάσια στην πλειοψηφία των περιπτώσεων αλλά είναι λίγο προβληματική σε προγράμματα τύπου «βάσεως δεδομένων» (data-base) που βασίζονται στη RAM. Η διαδικασία του νοικοκυρέματος και της μετακίνησης του σωρού μπορεί να χρειαστεί κάπου δύο λεπτά για να εκτελεσθεί, ανάλογα με το μέγεθός της. Αυτή η καθυστέρηση μπορεί να είναι ενοχλητική αλλά ευτυχώς υπάρχει μια απλή λύση. Οι buffers μπορούν να είναι μόνιμα κατανεμημένοι με τις ακόλουθες λίγες γραμμές προγράμματος:

```
10 OPENOUT «Buffer»
20 MEMORY νέα
30 CLOSEOUT
```

όπου «νέα» είναι η τωρινή τιμή της HIMEM που δίνεται από την ακόλουθη σχέση:

νέα = αρχική HIMEM - 4K - 1

Ο όρος -1 χρειάζεται για να εξασφαλιστεί ότι ο buffer της κασέτας που ανοίχτηκε στη γραμμή 10 δεν είναι αμέσως πάνω από την HIMEM όταν το αρχείο είναι κλειστό (δες παραπάνω). Η BASIC δεν έχει έτσι τη δυνατότητα να ανακτήσει τη μνήμη που έχει κατανεμηθεί στους buffers. Εν τούτοις, αν σε επόμενη φάση του προγράμματος, χρειαστούν οι buffers της κασέτας ο

χώρος που είχε κατανεμηθεί προηγουμένως θα ξαναχρησιμοποιηθεί. Σας παρουσιάζομε εδώ ένα πρακτικό παράδειγμα.

Αν υποθέσουμε μια τελείως τυπική συσκευή (χωρίς disk drives, οπτικά μολύβια (light pens), κ.λπ. η αρχική HIMEM θα είναι &AB7F. Αφαιρώντας 4K από τη HIMEM έχουμε &9B7F και με μία ακόμα ελάττωση κατά 1, για να συμμορφωθούμε με τα παραπάνω, παίρνουμε &9B7E. Κατά συνέπεια, οι buffers της κασέτας μπορούν να κατανεμηθούν μόνιμα σε έναν υπολογιστή με κασσέτα με τα παρακάτω:

```
10 OPENOUT «Buffer»
20 MEMORY &9B7E
30 CLOSEOUT
```

Μια εναλλακτική γραφή είναι:

```
10 OPENOUT «Buffer»
20 MEMORY HIMEM - 1
30 CLOSEOUT
```

Αν και το τελευταίο παράδειγμα μπορεί να φαίνεται προτιμότερο, έχει το σοβαρό μειονέκτημα ότι κρατούνται άλλα 4K κάθε φορά που «ξανατρέχει» ένα πρόγραμμα, με τις παραπάνω γραμμές. Χρησιμοποιώντας την προηγούμενη μέθοδο, είναι εύκολο να φυλάξουμε συγχρόνως ένα κομάτι μνήμης για ρουτίνες σε γλώσσα μηχανής. Για παράδειγμα αν απαιτούνται &200 (512 δεκαδικά) bytes για μια ρουτίνα σε γλώσσα μηχανής τότε η γραμμή 20 θα μπορούσε να αλλαχτεί σε:

```
20 MEMORY &997F
```

Η γλώσσα μηχανής μπορεί τότε να δουλέψει από το &9980 και πέρα. Συμπτωματικά αυτή είναι η περιοχή, όπου τοποθετούνται οι ρουτίνες ταξινόμησης σε γλώσσα μηχανής, που περιγράφονται στο Κεφάλαιο 5.

Αποθήκευση σε δίσκους

Αυτό το βιβλίο ασχολείται σχεδόν εξ ολοκλήρου με αρχεία σε κασέτα ή δίσκο. Εδώ ακολουθεί λοιπόν μια συζήτηση γι' αυτό το είδος συστήματος αποθήκευσης. Το πιο προφανές πλεονέκτημα συστήματος δίσκου σε σχέση με την κασέτα είναι η ταχύτητα μεταφορά δεδομένων. Αν και έχουμε επαινέσει την ταχύτητα της κασέτας του Amstrad είναι αναντίρρητο ότι παραμένει αργή σε σύγκριση με ένα καλά σχεδιασμένο σύστημα δίσκου. Εν τούτοις, αξίζει να αναφερθεί ότι η ταχύτητα ενός συστήματος δίσκων εξαρτάται σε μεγάλο βαθμό από το *λειτουργικό σύστημα δίσκου - Το DOS (disk operating system).* Αν το DOS, που είναι απλά ένα σύνολο σύνθετων προγραμμάτων δεν

έχει σχεδιαστεί προσεκτικά, η αναμενόμενη ταχύτητα του δίσκου δεν υλοποιείται και το σύστημα που προκύπτει θα μπορούσε, στη χειρότερη περίπτωση, να καταντήσει μόνο δυό ή τρεις φορές γρηγορότερο από ένα καλό σύστημα κασέτας. Αλλά, για να πούμε όλη την αλήθεια, η ταχύτητα δίσκου είναι μόνο μία από τις αρετές του. Όσον αφορά τα συστήματα αρχειοθέτησης το πιο εξέχον πλεονέκτημα του δίσκου έγκειται στη *φύση του, δηλ. της τυχαίας προσπέλασης (random access).* Αυτό σημαίνει ότι είναι δυνατό να πάτε κατ' ευθείαν (ή σχεδόν κατ' ευθείαν) στην εγγραφή ή στο αρχείο που ακριβώς θέλετε χωρίς να πρέπει να διαβάσετε όλα τα δεδομένα που δεν χρειάζεστε και που τυχαίνει να είναι πριν απ' αυτό. Κι αυτό γιατί οι κεφαλές ανάγνωσης, που παίρνουν τα δεδομένα από τον περιστρεφόμενο δίσκο, είναι στο άκρο ενός βραχίονα που μπορεί να κινείται ακτινικά πάνω στην επιφάνεια του δίσκου μέχρι να φτάσει την επιθυμητή τροχιά. Ο μηχανισμός που είναι υπεύθυνος για την ακτινική κίνηση λέγεται «κινητήρας βηματισμού» και, επειδή οι κινήσεις ελέγχονται ψηφιακά, μπορεί να τοποθετήσει την κεφαλή ανάγνωσης με μεγάλη ακρίβεια πάνω στο κέντρο μιας επιθυμητής τροχιάς. Είναι αυτή η ιδιότητα της τυχαίας προσπέλασης μαζί με την αυξημένη ταχύτητα περιστροφής του δίσκου, στην οποία οφείλεται η δημοτικότητα των αρχείων των βασισμένων σε δίσκους. Το DOS περιλαμβάνει συνήθως μια ομάδα εντολών, που εμφανίζονται σαν πρόσθετες εντολές (keywords) της BASIC, που επιτρέπουν στους χρήστες να επικοινωνούν με το δίσκο (Σ.τ.μ. Για περισσότερες λεπτομέρειες δες το βιβλίο «Το σύστημα δίσκων του Amstrad των εκδόσεων ΚΛΕΙΔΑΡΙΘΜΟΣ»).

Σημ.: Επειδή οι τύποι CPC664 και CPC6128 ενσωματώνουν από μόνοι τους μνήμη, για την προσωρινή αποθήκευση των αρχείων δεν απαιτείται η χρήση των εντολών:

```
OPENOUT "BUFFER"
MEMORY  &XXX
CLOSEOUT
```

Όλα τα προγράμματα δουλεύουν χωρίς μετατροπή της μεταβλητής HIMEM.

Περίληψη

1. Ένα σύστημα αρχειοθέτησης πρέπει να έχει μεγάλη χωρητικότητα αποθήκευσης και γρήγορη προσπέλαση σε οποιοδήποτε συγκεκριμένο στοιχείο του.
2. Η συνηθισμένη εσωτερική μνήμη πριν εισαχθεί η RAM με ημιαγωγούς ήταν η μνήμη μαγνητικών πυρήνων.
3. Μια μόνιμη εσωτερική μνήμη είναι εκείνη που διατηρεί τις αποθηκευμένες πληροφορίες μετά τη διακοπή της παροχής ρεύματος.
4. Η RAM με ημιαγωγούς είναι μη μόνιμη.
5. Η ταχύτητα ταινίας μετριέται συμβατικά σε baud. 1 baud είναι η ταχύ-

τητα ενός bit πληροφορίας ανά δευτερόλεπτο.

6. Κανονικά χρειάζονται 10 bits για να αντιστοιχηθεί κάθε χαρακτήρας σε μια ταινία.
7. Η Μεταγλώσσα είναι μία εξειδικευμένη ομάδα συμβόλων που χρησιμοποιείται για να περιγράφει τη συμπεριφορά των εντολών (keywords) της BASIC. Είναι περιεκτική και, σε αντίθεση με τις απλές γλωσσικές περιγραφές, χωρίς αμφιλογίες.
8. Η εντολή EVERY, επιτρέπει την πρόσβαση σε οποιοδήποτε από τους τέσσερις χρονομετρητές των interrupts. Η χρονική μονάδα είναι 1/50 του δευτερολέπτου.
9. Τα interrupts που δημιουργούνται από την EVERY μπορούν να αποενεργοποιηθούν με το DI και να επανεργοποιηθούν με το EI.
10. Για να ενεργοποιήσετε τη συλλογή άχρηστων χρησιμοποιείστε το FRE («»).
11. Η UPPER$ εξασφαλίζει όλο το κείμενο σε κεφαλαία· η LOWER$ εξασφαλίζει όλο το κείμενο σε μικρά.
12. Η αρχική ταχύτητα ταινίας είναι 1000 baud (SPEEDWRITE0). Η ταχύτητα μπορεί να διπλασιαστεί για οικονομία χρησιμοποιώντας την εντολή SPEEDWRITE1.
13. Το φόρτωμα του προγράμματος της ταινίας (LOAD) γίνεται αυτόματα με την ταχύτητα με την οποία σώθηκε (SAVED).
14. Βεβαιωθείτε ότι η αρχή της ταινίας έχει περάσει πριν αρχίσει κάποια εγγραφή.
15. Χρησιμοποιείτε ταινίες C-12 και, αν είναι δυνατό, βάζετε μόνο ένα αρχείο σε κάθε κασέτα.
16. Οι πληροφορίες πάνω σε μια ταινία, καταγράφονται σε μια σειρά από blocks (ομάδες). 1 block = 2K bytes.
17. Δύο περιοχές RAM, 2K κάθε μία, απαιτούνται για τους σκοπούς του buffer κασέτας.
18. Οι δύο περιοχές buffers της κασέτας στη RAM δεν είναι σε σταθερές διευθύνσεις. Κυμαίνονται δυναμικά πάνω και κάτω από τη HIMEM.
19. Στους buffers μπορούν να δοθούν σταθερές διευθύνσεις, μέσω μιας άμεσης ανάθεσης (direct assignment).

Ασκήσεις

1.1. Ποιό πλεονέκτημα είχε η παλιά μνήμη μαγνητικού πυρήνα σε σχέση με τη σύγχρονη RAM με ημιαγωγούς;

1.2. Ποιά είναι η διαφορά ανάμεσα στη μνήμη και στην περιφερειακή μνήμη (store);

1.3. Ποιά είναι η σχέση ανάμεσα στο Κάνσας Σίτυ και το κασετόφωνο;

1.4. Πόσοι χαρακτήρες υπάρχουν σε ένα μπλόκ;

1.5. Γιατί υπάρχουν δύο περιοχές για buffers ταινίας των 2K η κάθε μία στη μνήμη;

1.6. Η EOF είναι μία ψευδομεταβλητή που περιέχει είτε 0 είτε -1. Ποιά εί-

ναι η σημασία της όταν περιέχει -1;

1.7. Ποιά είναι η διαφορά ανάμεσα στη FRE (0) και τη FRE («»);

1.8. Μια ταινία γραμμένη στα 2000 baud μπορεί να αναγνωστεί χωρίς σφάλμα από τη συσκευή στην οποία σώθηκε αλλά μπορεί να μην διαβαστεί σε άλλη συσκευή Γιατί;

Κεφάλαιο Δύο

Συνθετικά Στοιχεία ενός Συστήματος Διαχείρισης Αρχείων

Σχεδιασμός αρχείου

Ανέκαθεν η επιστήμη έδινε έμφαση στην αξία της ταξινόμησης. Στην πραγματικότητα, η επιστήμη θα μπορούσε να οριστεί σαν συστηματική τακτοποίηση εξακριβωμένων γνώσεων. Δε μας αφορά, τη στιγμή αυτή, η διαφορά ανάμεσα στις «εξακριβωμένες» γνώσεις και τις πληροφορίες παρά μόνο για να επισημάνουμε ότι η πληροφορία μπορεί, μερικές φορές, να είναι λανθασμένη. Όσον αφορά όμως την οργάνωση και την τακτοποίηση του αρχείου, δεν έχει σημασία αν ένα οποιοδήποτε στοιχείο στο αρχείο περιέχει αληθινές ή λανθασμένες πληροφορίες. Η ακρίβεια των πληροφοριών που κρατιούνται στο αρχείο εξαρτάται από αυτούς που είναι υπεύθυνοι για τη διαχείρισή τους και την καθημερινή ενημέρωσή τους. Εν πάσει περιπτώσει, ο σχεδιασμός ενός συστήματος διαχείρισης αρχείων πάντα θα έπρεπε να λαβαίνει υπ' όψη του την πιθανότητα λανθασμένων εισαγωγών και να εξασφαλίζει ότι τα απλά λάθη ανακαλύπτονται εύκολα στο στάδιο της εισόδου και, εξ ίσου εύκολα, διορθώνονται.

Θα πρέπει να γίνει κατανοητό ότι τα προγράμματα που οργανώνουν και επεξεργάζονται πληροφορίες δεν είναι παρά εργαλεία. Η πρακτική αξία ενός συστήματος διαχείρισης αρχείων σε υπολογιστή εξαρτάται σε μεγάλο βαθμό από τον τρόπο με τον οποίο έχουν ταξινομηθεί οι πληροφορίες του αρχείου. Ο υπολογιστής δε θα τις ταξινομήσει. Ένα καλό πρόγραμμα υπολογιστή κανονικά θα επιτρέψει στο χρήστη αξιοσημείωτη ελευθερία στον τρόπο δόμησης των πληροφοριών του αρχείου αλλά, τελικά, είναι ευθύνη του χρήστη να αποφασίσει τον τρόπο που θα χρησιμοποιήσει αυτή την ελευθερία. Για παράδειγμα, το όνομα που δίνεται σε κάθε επικεφαλίδα στο αρχείο, το υλικό που θα συμπεριληφθεί, ο αριθμός των διαφόρων επικεφαλίδων, ο χώρος που διατίθεται για κάθε στοιχείο κειμένου κάτω από μια επικεφαλίδα, οι συντομογραφίες που θα χρησιμοποιηθούν, η ιδιαίτερη επικεφαλίδα που θα θεωρηθεί σημαντικότερη από άλλες επικεφαλίδες —όλα αυτά τα σημεία και πολλά άλλα πρέπει να εξεταστούν προσεκτικά πριν αποφασιστεί ο αρχικός σχεδιασμός του αρχείου. Είναι ουσιώδες να υιοθετηθεί μια μακροπρόθεσμη προοπτική στη διάρκεια του σταδίου σχεδιασμού. Θα ήταν α-

νοησία, ίσως καταστροφή, αν ένα αρχείο έχει λειτουργήσει για κάποιο χρονικό διάστημα και ξαφνικά διαπιστωθεί ότι μια πρόσθετη επικεφαλίδα θα 'πρεπε να είχε εισαχθεί στη διάρκεια της αρχικής φάσης σχεδιασμού. Απαιτείται χρόνος και εργασία για τη διατήρηση ενός αρχείου ενημερωμένου με νέες ή διορθωμένες πληροφορίες, έτσι το να το διαλύσουμε και να το ξαναεισάγουμε πάλι ολόκληρο σε διαφορετική διάταξη θα ήταν μια δυσάρεστη και χρονοβόρα εμπειρία.

Είναι βέβαια δυνατό να προλάβουμε ένα τέτοιο ατύχημα, δημιουργώντας απλώς μια εφεδρική επικεφαλίδα που να μείνει κενή μέχρις ότου, και όταν, χρειαστεί. Αυτό είναι εν τάξει αν υπάρχει άφθονος χώρος στην περιφερειακή μνήμη αλλά δεν είναι παρά ένα μπάλωμα για ένα πιθανό πρόβλημα που δε θα προέκυπτε αν το αρχικό αρχείο είχε σχεδιαστεί πιο προσεκτικά. Είναι επίσης δυνατό να σχεδιάσουμε ένα πρόγραμμα διαχείρισης αρχείων, έτσι που να επιτρέπει να προστίθενται νέες επικεφαλίδες, ή να εξαλείφονται οποτεδήποτε θέλουμε, αλλά αυτό θα συνεπάγονταν την προσθήκης άλλης μιας ακόμα διάστασης ευκαμψίας. Πραγματικά, ένα πρόγραμμα χειρισμού αρχείων θα μπορούσε να σχεδιαστεί για να προβλέπει οποιεσδήποτε ελλείψεις κατά την κρίση του χρήστη αλλά τελικά το πρόγραμμα θα αποκτούσε δυσκολομεταχείριστες ιδιότητες και θα κράταγε πάρα πολύ διαθέσιμο χώρο στη RAM.

Φιλικότητα προς το χρήστη

Η «Φιλικότητα προς το χρήστη» είναι ένα αμφισβητήσιμο ζήτημα, έτσι δεν προκαλεί έκπληξη το ότι έχουν γραφτεί τόσα πολλά γι' αυτήν. Για μερικούς, είναι σχεδόν ένα είδος λατρείας. 'Αλλοι, ενώ αναγνωρίζουν τη σημασία της, αισθάνονται ότι η αξία της έχει υπερτιμηθεί κατά πολύ. Το παρακάτω προτείνεται μάλλον σαν σημείο εκκίνησης για συζήτηση παρά σαν τυπικός ορισμός:

Φιλικότητα προς το χρήστη είναι ένα μέτρο της ικανότητας ενός προγράμματος να αναγνωρίζει την ανθρώπινη αδυναμία.

Οι πρόσθετες γραμμές προγράμματος που απαιτούνται για την προσθήκη κάποιου βαθμού φιλικότητας προς το χρήστη λέγονται συχνά (γραμμές) *ελέγχου ηλιθιότητας*. 'Ολοι μας είναι πιθανόν να παρουσιάζουμε σε κάποιες στιγμές ηλίθια συμπεριφορά έτσι δε θα 'πρεπε να θεωρήσουμε τον όρο «ηλιθιότητα» σαν προσβλητικό. Το ουσιώδες ζήτημα εδώ είναι να αποφασίσουμε πόσος έλεγχος ηλιθιότητας δικαιολογείται σε ένα πρόγραμμα διαχείρισης αρχείων.

Υπάρχουν αρκετοί τρόποι για να μπούμε στο θέμα αλλά η παρακάτω σειρά είναι τυπική:

(1) Ο υπολογιστής ζητάει κάποια στοιχεία εισόδου.
(2) Ο χειριστής τα εισάγει.

(3) Το πρόγραμμα εξετάζει τα στοιχεία εισόδου και είτε

(α) τα δέχεται, είτε
(β) τα απορρίπτει και ζητάει ξανά στοιχεία εισόδου.

Η διαδικασία επαναλαμβάνεται επ' αόριστον μέχρι τα στοιχεία εισόδου να ικανοποιήσουν τον υπολογιστή. Μία τυπική διαδικασία ελέγχου αξιοπιστίας των στοιχείων εισόδου δείχνεται στο παρακάτω παράδειγμα.

Παράδειγμα:

```
100 INPUT"ENTER NUMBER OF RECORDS";N%
120 IF N%<1 OR N%>500 THEN 100
```

Το πάνω όριο των 500 φυσικά είναι αυθαίρετο. Το πρόγραμμα κάνει εν μέρει μόνο έλεγχο - ηλιθιότητας γιατί δεν διαφωτίζει το χειριστή που δεν ξέρει, γιατί απορρίφτηκε το στοιχείο εισόδου. Χρειάζεται λοιπόν ένα μήνυμα από τον υπολογιστή που να εξηγεί το γιατί. Το πρόγραμμα τώρα μεγαλώνει λίγο:

```
100 INPUT"ENTER NUMBER OF RECORDS";N%
120 IF N%<1 OR N%>500 THEN PRINT"NUMBER
MUST BE IN RANGE (1 TO 500)":GOTO 100
130 PRINT "NUMBER ACCEPTED"
```

Μία άλλη άποψη της φιλικότητας προς το χρήστη είναι η ενσωμάτωση μηνυμάτων «Είσαι βέβαιος; Απάντησε ΝΑΙ/ΟΧΙ (YES/NO). Αυτά μπαίνουν σε περιπτώσεις όπου η λανθασμένη απάντηση του χειριστή θα μπορούσε να προκαλέσει σοβαρό πρόβλημα. Για παράδειγμα, το αρχικό ερώτημα θα μπορούσε να ήταν «Θέλεις αυτή η εγγραφή να σβηστεί;» Αν ο χειριστής, σε μια στιγμή αμέλειας απαντήσει «Y» (αλλά ήθελε «N») μια πολύτιμη εγγραφή θα μπορούσε να χαθεί για πάντα. Αυτό θα ήταν καταστροφικό έτσι το επόμενο μήνυμα, «Είσαι βέβαιος;» θα βοηθούσε να μειωθεί ένας τέτοιος κίνδυνος. Για να έχετε μια πρόσθετη προστασία, ενδείκνυται να κάνετε το πρόγραμμα να απαιτεί το πάτημα ενός συγκεκριμένου πλήκτρου για την πιο επικίνδυνη εναλλακτική δυνατότητα ενώ σε κάθε άλλη περίπτωση να δέχεται οποιοδήποτε από τα άλλα.

Η ιδέα της φιλικότητας προς το χρήστη καλύπτει ένα πεδίο ευρύτερο από τον απλό έλεγχο - ηλιθιότητας. Τα απλά και σαφή μηνύματα στην οθόνη και η σωστή εκλογή χρωμάτων για έμφαση είναι επίσης σημαντικά. Το χρώμα σαν αυτοσκοπός δε δικαιολογείται αν και ο Amstrad προσφέρει μια πλατιά γκάμα χρωμάτων. Τα αλλόκοτα χρωματικά εφφέ σε προγράμματα παιχνιδιών είναι χρήσιμα σα διακοσμητικά βοηθήματα αλλά, σε πιο σοβαρά προγράμματα, το χρώμα πρέπει να χρησιμοποιείται μόνο σε περιπτώσεις όπου μπορεί να βελτιώσει ή να ενισχύσει τα εμφανιζόμενα στην οθόνη.

Μερικές πλευρές της φιλικότητας προς το χρήστη μπορεί να φαίνονται

εξοργιστικές. Για ν' αρχίσουμε, θα μπορούσαμε να ρωτήσουμε για ποιό συγκεκριμένο χρήστη πρόκειται να είναι φιλικό το πρόγραμμα. Ένα πρόγραμμα που χάρη σε άφθονα προειδοποιητικά μηνύματα, θα μπορούσε να θεωρηθεί «φιλικό» από ένα αρχάριο χειριστή, μπορεί να ταλαιπωρήσει έναν έμπειρο χειριστή. Χρειάζεται κάποιος χρόνος για οποιοδήποτε χειριστή για να μάθει πως να κάνει χρήση όλων των ευκολιών που προσφέρει ένα έξυπνο πρόγραμμα, έτσι, αρχικά, πρέπει να εκτιμηθεί η φιλικότητα προς το χρήστη. Αλλά ο ίδιος ο αρχάριος μπορεί τελικά να γίνει έμπειρος και να αρχίσει να εκνευρίζεται αν πάρα πολλά προειδοποιητικά μηνύματα πρέπει να απαντηθούν με εισαγωγές «Y/N». Ένας άλλος λόγος για να μην προγραμματίζουμε με πάρα πολλή φιλικότητα προς το χρήστη είναι το κόστος σε χώρο RAM. Είναι εύκολο να το παρακάνουμε προσπαθώντας να προβλέψουμε όλες τις πιθανές μορφές ανθρώπινου λάθους. Το αποτέλεσμα θα μπορούσε να είναι τελικά μια κατάσταση, όπου η επιθυμία για φιλικότητα ικανοποιείται μόνο θυσιάζοντας χρήσιμες δυνατότητες. Τα παραδείγματα προγραμμάτων σ' αυτό το βιβλίο είναι «μετρίως» φιλικά αλλά, επειδή έχει γίνει μεγάλη χρήση *τυποποιημένων* προγραμμάτων (υπορουτινών), θα ήταν εύκολο για τον καθένα να προσθέτει επί πλέον ελέγχους εισόδου δεδομένων χωρίς να βάζει σε κίνδυνο την όλη δομή του προγράμματος.

Σπάσιμο του προγράμματος

Όταν ένα πρόγραμμα σταματάει να είναι υπό τον έλεγχο του χειριστή λέμε ότι έχει *σπάσει*. Είναι ένας πολύ πλατύς ορισμός γιατί υπάρχουν διάφοροι τρόποι, με τους οποίους μπορεί να σπάσει ένα πρόγραμμα. Το σπάσιμο ενός προγράμματος είναι πάντα ενοχλητικό γιατί σημαίνει να ξαναρχίσουμε πάλι με ένα RUN ή ακόμα και με μία επαναφόρτωση από την περιφερειακή μνήμη. Στην περίπτωση προγραμμάτων που επεξεργάζονται μεγάλα ποσά δεδομένων, ένα σπάσιμο μπορεί να είναι ενοχλητικό, ιδιαίτερα με αρχεία βασισμένα στη RAM, αν συμβεί κοντά στο τέλος μιας μακράς σειράς εισαγόμενων δεδομένων. Τα βήματα που ακολουθούνται στο πρόγραμμα για να αποφευχθεί ένα σπάσιμο από υπαιτιότητα του χειριστή είναι μια ακόμα πλευρά της φιλικότητας προς τον χρήστη.

Γραπτή τεκμηρίωση (documentation)

Όλα τα προγράμματα, εκτός από τα απλούστερα, χρειάζονται κάποια μορφή υποστήριξης με γραπτή τεκμηρίωση που να εξηγεί πως μπορούν να χρησιμοποιηθούν αποτελεσματικά. Η καλή γραπτή τεκμηρίωση είναι ιδιαίτερα αναγκαία σε προγράμματα διαχείρισης αρχείων, γιατί αυτά κατά κανόνα προσφέρουν ένα πλατύ φάσμα δυνατοτήτων επεξεργασίας που μπορεί να φανούν πολύ περίπλοκες σ' αυτόν που τις χρησιμοποιεί για πρώτη φορά. Η καλή γραπτή τεκμηρίωση συμβάλλει επίσης στη φιλικότητα προς το χρήστη μειώνοντας την εξάρτηση από τα μηνύματα οθόνης. Οι πάρα πολλές πληροφορίες από την οθόνη, αν και αρχικά παρέχουν πολύτιμα στοιχεία, γρήγορα εκφυλίζονται σε άχρηστη πολυλογία καθώς ο χειριστής γίνεται

πιο έμπειρος. Η συνεχής υπενθύμιση με κάθε εμφάνιση σελίδας ότι πρέπει να «Πατήσουμε το Πλήκτρο X για να ξαναγυρίσουμε στις Επιλογές» μπορεί γρήγορα να γίνει εκνευριστική. Πέρα απ' αυτό, σπαταλάει την περιοχή της οθόνης. Πληροφορίες αυτού του είδους δίνονται πολύ καλύτερα από κάποια γραπτή τεκμηρίωση.

Μερικοί προκαταρκτικοί όροι και ορισμοί

Μέχρι αυτό το σημείο, έχουμε χρησιμοποιήσει όρους όπως «πληροφορία», «δεδομένα», «αρχείο», κ.λπ., με την καθημερινή τους έννοια. Έχουμε φτάσει τώρα στο σημείο, όπου για λόγους διάκρισης, χρειάζονται πιο τυποποιημένοι ορισμοί.

Δεδομένα (data)

Από όλους τους όρους στο λεξιλόγιο της γλώσσας των υπολογιστών ο όρος «δεδομένα» είναι πιθανόν ο πιο πολυχρησιμοποιημένος. Με τη γενική έννοια, σαν δεδομένο θα μπορούσε να οριστεί οτιδήποτε έχει νόημα για τον υπολογιστή. Αυτός ο ορισμός είναι κάτι που επίσης τα αγκαλιάζει όλα για τους δικούς μας σκοπούς. Αυτό που χρειαζόμαστε είναι κάποιος κανόνας από τον οποίο να μπορούμε να ξεχωρίζουμε το «δεδομένο» από άλλους όρους, με τους οποίους έχει στενή σχέση. Θα χρησιμοποιήσουμε τον ακόλουθο απλό ορισμό:

> Δεδομένα (data) είναι πληροφορίες που μπορούν να κρατηθούν στην περιφερειακή μνήμη, στα οποία έχουμε πρόσβαση με ένα κατάλληλο πρόγραμμα.

Όταν τα δεδομένα φορτώνονται από την περιφερειακή μνήμη στη RAM είναι, φυσικά, ακόμα «δεδομένα» αλλά μόνο με τη γενική έννοια.

Αρχεία (Files)

Όπως τα δεδομένα, ο όρος «αρχείο» είναι επίσης ένας πολυδουλεμένος όρος και χρησιμοποιείται συχνά με μια ποικιλία σημασιών. Για παράδειγμα συνηθίζεται σε οδηγίες χρήσης να ονομάζεται «αρχείο» οτιδήποτε αποθηκεύεται σε ταινία ή δίσκο. Αναφέρονται σε «Αρχεία προγραμμάτων» και «Αρχεία δεδομένων». Σ' αυτό το βιβλίο θα ορίσουμε το αρχείο με τον ακόλουθο τρόπο:

> Ένα αρχείο είναι μια συλλογή σχετικών μεταξύ τους δεδομένων που φυλάγονται στην περιφερειακή μνήμη και μπορούν να προσπελαθούν από ένα πρόγραμμα της RAM.

Για παράδειγμα, τα δεδομένα σ' ένα αρχείο θα μπορούσαν να αποτελού-

νται από διάφορα στοιχεία πληροφοριών για πουλιά. Ένα άλλο αρχείο θα μπορούσε να είναι ένας απλός κατάλογος ονομάτων, διευθύνσεων, και αριθμών τηλεφώνων πελατών.

Όπως εξηγήθηκε νωρίτερα, η εκλογή του υλικού που θα πάει σ' ένα αρχείο είναι μια ανθρώπινη απόφαση. Το αν τα περιεχόμενα του αρχείου είναι πραγματικά σχετικά μεταξύ τους ή όχι πρέπει να εξαρτάται από την κρίση του ατόμου που δημιουργεί το αρχείο.

Σύμφωνα με τον προηγούμενό μας ορισμό των δεδομένων, έπεται ότι ο όρος «αρχείο» προϋποθέτει ότι αυτό είναι ένα αρχείο δεδομένων. Όπως θα γνωρίζουν οι περισσότεροι αναγνώστες, τα προγράμματα σε BASIC αποθηκεύονται με μία ειδική μορφή (format) με τη χρήση *λημμάτων* (tokens) αντί των λέξεων της BASIC ενώ τα αρχεία δεδομένων αποθηκεύονται σε απλούς χαρακτήρες ASCII.

Εγγραφή (Record)

Αν και ένα αρχείο περιέχει πληροφορίες που είναι κατά κάποιο τρόπο σχετικές μεταξύ τους, έχει πάντως μια ιεραρχημένη μορφή. Αυτό σημαίνει ότι θα αποτελείται από μια ομάδα ατομικών «εγγραφών». Για παράδειγμα, ένα αρχείο ονομάτων και διευθύνσεων πελατών θα περιέχει ιδιαίτερη εγγραφή για κάθε πελάτη. Μπορούμε συνεπώς να ορίσουμε την εγγραφή ως εξής:

> Μια εγγραφή περιέχει όλα τα δεδομένα τα σχετικά με μία ιδιαίτερη καταχώρηση.

Για παράδειγμα, ένα αρχείο για τα πτηνά της Βρετανίας θα μπορούσε να περιέχει ένα αριθμό εγγραφών, μία για κάθε διαφορετικό πτηνό.

Πεδίο (Field)

Το «πεδίο» είναι για την εγγραφή ό,τι η εγγραφή είναι για το αρχείο, έτσι κατάλληλος είναι ο παρακάτω ορισμός:

> Πεδίο είναι μια υποδιαίρεση της εγγραφής και αντιπροσωπεύει ένα τμήμα του συνόλου των πληροφοριών μιας εγγραφής

Παίρνοντας, σαν παράδειγμα, ένα αρχείο των ηφαιστείων του κόσμου, ένα πεδίο θα μπορούσε να δίνει το όνομα του ηφαίστειου, ένα άλλο το ύψος και ένα τρίτο τη θέση του. Ο αριθμός των χαρακτήρων που κρατούνται για κάθε πεδίο είναι γνωστός σαν χώρος πεδίου (field - width).

Επικεφαλίδες πεδίου

Όλα τα πεδία πρέπει να έχουν μια *επικεφαλίδα* για να έχουν νόημα τα δεδομένα. Για παράδειγμα, αν ο αριθμός 5.67 εμφανιζόταν σε ένα πεδίο μιας εγγραφής, δε θα είχε νόημα χωρίς μια επικεφαλίδα. Μια επικεφαλίδα πεδίου,

ας πούμε, «Ύψος σε μέτρα» δίνει νόημα στον αριθμό. Όσον αφορά την αποτελεσματικότητα του προγραμματισμού, ίσως είναι σημαντικό να κάνουμε τη διάκριση ανάμεσα στα πεδία που φέρουν αποκλειστικά *αριθμητικά* δεδομένα και σ' εκείνα που φέρουν *αλφαριθμητικούς* χαρακτήρες. Κατά τη δημιουργία ενός νέου αρχείου ο χειριστής μπορεί να αναρωτηθεί τι είδους δεδομένα να καταχωρεί κάτω από κάθε επικεφαλίδα πεδίου, αλφαριθμητικά ή αριθμητικά; Είναι εύκολο να υποθέτουμε ότι όλα τα πεδία είναι σε αλφαριθμητική μορφή και να βάλουμε τον υπολογιστή να τα ξεδιαλέξει, αλλά είναι αποτελεσματικότερο τα αριθμητικά πεδία να είναι διακεκριμένα από τα αλφαριθμητικά κατά την εισαγωγή τους από το πληκτρολόγιο. Ιδιαίτερα μπορούν να χρησιμοποιηθούν ακέραιοι, αντί για αριθμούς με κινητή υποδιαστολή.

Η σημασία του Πεδίου Κλειδιού (Key field)

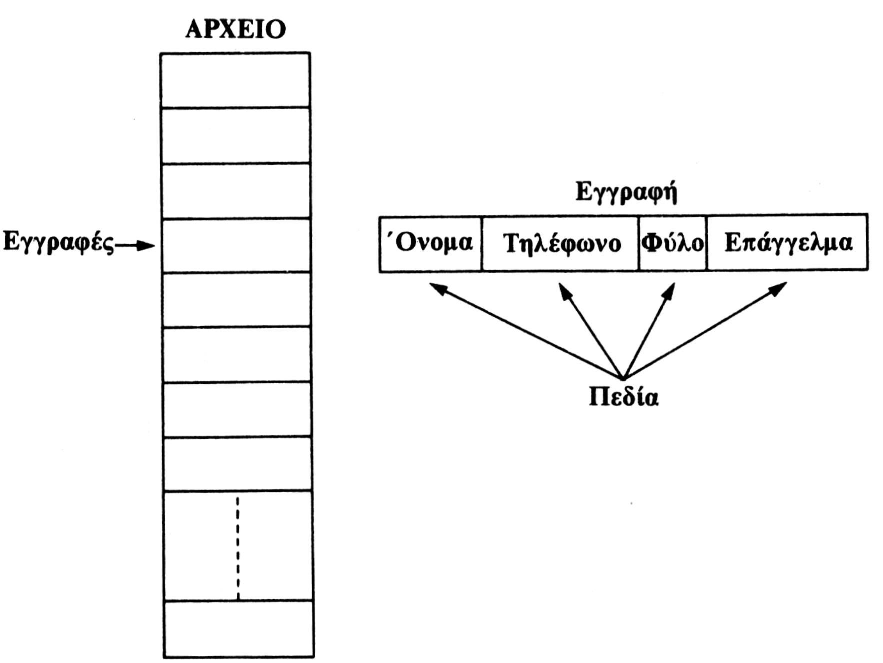
Αν και όλα τα πεδία μιας εγγραφής θα περιέχουν σημαντικές πληροφορίες, ένα από αυτά θα έχει υψηλότερο κύρος από τα άλλα. Λέγεται *πεδίο-κλειδί,* γιατί χρησιμοποιείται για να αναγνωρίζεται η εγγραφή. Την ευθύνη να αποφασίσει ποιό συγκεκριμένο πεδίο θα εκλεγεί σαν *κλειδί* την έχει το άτομο που αρχικά δημιουργεί το αρχείο. Πάντως, επειδή το πεδίο-κλειδί αποτελεί την *ταυτότητα* της εγγραφής, είναι ουσιώδες να είναι απολύτως μονοσήμαντο. Στο παράδειγμά μας του αρχείου ηφαιστείων, σαν εκλογή πεδίου-κλειδιού σαφώς θα μπορούσε να είναι το όνομα του ηφαιστείου, αφού δεν θα υπήρχαν δύο ηφαίστεια με το ίδιο όνομα. Από την άλλη μεριά, το ύψος του ηφαιστείου ίσως να μην είναι πάντα μονοσήμαντο. Σε μερικά αρχεία θα δημιουργόνταν προβλήματα, ιδιαίτερα σ' εκείνα όπου πεδίο-κλειδί είναι το όνομα του πελάτη. Είναι πολύ πιθανό να παρουσιάζονται περισσότεροι από ένας π.χ. Παπαδόπουλος στο ίδιο αρχείο. Αν συμβεί αυτό, θα μπορούσε να προστεθεί ένας πρόσθετος δείκτης-ίσως Παπαδόπουλος2. Η ασφαλής μέθοδος, ιδιαίτερα σε μεγαλύτερα αρχεία, είναι να χρησιμοποιούμε σαν πεδίο-κλειδί έναν κωδικό αναγνώρισης (code identifier) (ίσως δέκα ή περισσότερων χαρακτήρων) και όχι το όνομα ενός ατόμου. Οι κωδικοί αναγνώρισης φαίνονται απάνθρωποι και δυσάρεστοι σε πολλούς αλλά, επειδή είναι αναγκαίοι για τη διαχείριση μιας πολυπληθούς κοινωνίας, δεν θα 'πρεπε να θεωρούνται από τη σκοπιά του Όργουελ.

Αρχεία που ασχολούνται με καταγραφές εμπορευμάτων θα έχουν σχεδόν σίγουρα σαν πεδίο-κλειδί κάθε εγγραφής τον κωδικό αριθμό του εμπορεύματος. Ένας συνηθισμένος οίκος ηλεκτρονικών εξαρτημάτων θα έχει πιθανόν μια ξεχωριστή εγγραφή για κάθε αποθηκευμένο εξάρτημα. Ο αριθμός των διαφορετικών εξαρτημάτων μπορεί να είναι δεκάδες, ή ακόμα εκατοντάδες, χιλιάδες. Θα ήταν χρονοβόρα και δύσχρηστη διαδικασία το να προσπαθείς να βρείς μια μονοσήμαντη φυσική περιγραφή για κάθε πεδίο-κλειδί, έτσι ο αριθμός τεμαχίου είναι η προφανής λύση. Αν και χρησιμοποιήσαμε ένα παράδειγμα από τη βιομηχανία δεν είναι ασυνήθιστο να βρείτε μεγάλες ποσότητες κάποιου είδους σε σπίτια, ιδιαίτερα αν ένας απ' τους

ενοίκους είναι μανιώδης συλλέκτης. Για παράδειγμα, μια συλλογή γραμματοσήμων 10.000 κομματιών θα θεωρούνταν άνετα μεσαίου μεγέθους από οποιοδήποτε ερασιτέχνη φιλοτελιστή και ένα σύστημα διατήρησης αρχείου σε υπολογιστή θα έδινε μια άλλη διάσταση στο χόμπυ. Αν υποτεθεί ότι η συλλογή είναι κατάλληλα οργανωμένη σε αρχεία, ο υπολογιστής μπορεί να χρησιμοποιηθεί για να βρεί απροσδόκητες σχέσεις ανάμεσα σε ομάδες κομματιών. Ακόμη, ίσως να είναι βολικότερο, το πεδίο - κλειδί να είναι ένας αριθμός καταλόγου γραμματοσήμων παρά μια περιγραφή.

Τα αριθμητικά ή κωδικά πεδία-κλειδιά, παρά τα πλεονεκτήματά τους, τείνουν να μην είναι φιλικά προς το χρήστη. Είναι ευκολότερο να θυμάσαι τη «Μαύρη πέννα» ας πούμε, παρά τον αριθμό «23578642» όταν ψάχνεις για μια εγγραφή. Για να συνδυάσουμε τα πλεονεκτήματα ενός κλειδιού κωδικοποιημένης αναζήτησης με τη φιλικότητα ενός λεκτικού κλειδιού, μπορεί να χρησιμοποιηθεί μια έξυπνη διέξοδος γνωστή σαν *μέθοδος των δεικτών (indexing).*

Το σχήμα 2.1 επεξηγεί τη σχέση ανάμεσα στο πεδίο, την εγγραφή και το αρχείο.



(Σχ. 2.1.) Συστατικά ενός αρχείου.

Μέγεθος αρχείου

Διάφοροι παράγοντες μπορούν να επηρεάσουν τις πληροφορίες που ο-

μαδοποιούνται κάτω από μία επικεφαλίδα αρχείου, ανάμεσά τους και προσωπικές προτιμήσεις. Πέρα και πάνω απ' αυτό, θα υπάρχουν φυσικοί περιορισμοί που καθορίζει η μνήμη και το σύστημα αποθήκευσης ως προς τον μέγιστο επιτρεπόμενο αριθμό bytes σε ένα μόνο αρχείο υπολογιστή. Αυτοί οι περιορισμοί θα εξαρτιώνται σε μεγάλο βαθμό από τις μεθόδους που μεταχειρίζεται το πρόγραμμα. Υπάρχουν αρκετές αναγνωρισμένες μέθοδοι οργάνωσης προγραμμάτων διαχείρισης αρχείων και στη συνέχεια θα ασχοληθούμε με μερικές απ' αυτές. Από την άποψη του μήκους του αρχείου, είναι αρκετό να διακρίνουμε δύο μεγάλες κατηγορίες αρχείων:

(α) *Σειριακά αρχεία (serial files) βασισμένα στη RAM*: ολόκληρο το αρχείο πρέπει να φορτωθεί από την περιφερειακή μνήμη στη RAM, για να μπορέσει να γίνει πρόσβαση σε εγγραφές.
(β) *αρχεία βασισμένα στην περιφερειακή μνήμη*: το αρχείο παραμένει στην περιφερειακή μνήμη και όταν χρειάζεται, καλείται μόνο η επιλεγείσα εγγραφή, ή σε μερικές περιπτώσεις λίγες εγγραφές.

Απ' αυτό, γίνεται φανερό ότι το μήκος ενός αρχείου βασισμένου στη RAM περιορίζεται μόνο από την ποσότητα του χώρου που υπάρχει στη RAM πέρα και πάνω από εκείνη που χρησιμοποιεί το πρόγραμμα. Ένα πρόγραμμα με εκτεταμένες δυνατότητες επεξεργασίας αρχείου θα μπορούσε να καταλάβει μεγάλο χώρο στη RAM, αναγκάζοντας μας κατά συνέπεια να κάνουμε κάποιο συμβιβασμό ανάμεσα στις ικανότητες επεξεργασίας και το χώρο αρχείου. Αν υποθέσουμε ότι απαιτούνται 100 χαρακτήρες (περιλαμβανομένων των χαρακτήρων ελέγχου) για μία συνηθισμένη εγγραφή, το μέγιστο μήκος αρχείου δεν θα μπορούσε να υπερβαίνει τις 100 εγγραφές. Θα υπάρχει μία φυσική επιθυμία να συσσωρεύσουμε όσο γίνεται περισσότερες πληροφορίες μέσα σε κάθε εγγραφή έχοντας πολλά πεδία καθώς και χώρο για χαρακτήρες μέσα σε κάθε πεδίο. Η ακόλουθη ισότητα για τον υπολογισμό του μεγέθους του αρχείου σε bytes θα 'πρεπε να μας επηρεάσει σοβαρά στη διάρκεια της αρχικής φάσης σχεδιασμού:

Σύνολο bytes = Εύρος πεδίου Χ αριθμός πεδίων Χ αριθμός εγγραφών

Δεν εκτιμάται όμως πάντα ότι απαιτούνται εβδομήντα χαρακτήρες, περιλαμβανομένων των κενών, για το μέσο όνομα, διεύθυνση, ταχυδρομικό κώδικα και αριθμό τηλεφώνου.

Με αρχεία βασισμένα στην περιφερειακή μνήμη, το ποσό της διαθέσιμης RAM έχει μικρότερη σημασία γιατί ο περιοριστικός παράγοντας στο μήκος αρχείου είναι το ποσό της διαθέσιμης περιφερειακής μνήμης κατά τη λειτουργία. Εκτός από την αύξηση του μέγιστου μήκους αρχείου, ένα αρχείο βασισμένο σε περιφερειακή μνήμη αφήνει περισσότερο χώρο της RAM για το πρόγραμμα, έτσι ο αριθμός των λειτουργιών επεξεργασίας του αρχείου δεν χρειάζεται να περικοπεί τόσο δραστικά, όσο στα αρχεία τα βασισμένα στη RAM.

Παρ' όλο που έχουν μερικά μειονεκτήματα, τα αρχεία που είναι βασι-

σμένα στη RAM είναι χρήσιμα. Σε περιπτώσεις όπου το μήκος των αρχείων είναι σαφώς μέσα στις δυνατότητες της RAM, υπάρχει ένα ευδιάκριτο πλεονέκτημα κατά τη χρήση τους. Όταν φορτωθεί ένα πλήρες αρχείο από την περιφερειακή μνήμη, οι ουσιώδεις επεξεργασίες εκτελούνται με την ταχύτητα της RAM. Η ταξινόμηση εγγραφών με κάποιο κριτήριο είναι μια διαδικασία στην οποία τα αρχεία τα βασισμένα στη RAM υπερτερούν. Είναι όμως δυνατό, αλλά σχετικά αργό και άβολο, να ταξινομηθούν εγγραφές αν όλο το αρχείο δεν έχει μπει στη RAM.

Διάσπαση και σύζευξη αρχείων.

Ο αριθμός των εγγραφών κάτω από μια επικεφαλίδα αρχείου μπορεί τελικά να γίνει πολύ μεγάλος ώστε να μη χωράει στη RAM με ένα πέρασμα. Αυτό μπορεί να ξεπεραστεί με μια λειτουργία για τη διάσπαση ενός και μόνον αρχείου σε ένα ή περισσότερα χωριστά αρχεία. Για παράδειγμα, αν πεδίο-κλειδί είναι το όνομα κάποιου πράγματος, συνίσταται να διασπάσουμε το αρχείο σε ένα με ονόματα που αρχίζουν από A μέχρι J, ένα άλλο με ονόματα από K μέχρι R και ένα τρίτο από S μέχρι Z. Αντίστροφα, μπορεί να χρειαστεί να έχουμε μια δυνατότητα *σύζευξης (merging)* δύο ή περισσότερων μικρότερων αρχείων σε ένα.

Επιλογές επεξεργασιών

Το εξέχον πλεονέκτημα ενός υπολογιστή είναι η ικανότητά του να επεξεργάζεται δεδομένα με ποικίλους τρόπους. Ο τρόπος με τον οποίο οι διάφορες επεξεργασίες επιλέγονται και ενεργοποιούνται καθορίζει αν το πρόγραμμα ορίζεται σαν *κατευθυνόμενο από κατάλογο επιλογών (menu - driven) ή κατευθυνόμενο από εντολές (command - driven)*.

Ένα πρόγραμμα κατευθυνόμενο από κατάλογο επιλογών περιστρέφεται γύρω από μία οθόνη που αναφέρεται σαν «σελίδα καταλόγου» (menu page) που περιέχει τον κατάλογο των δυνατών επιλογών. Η απαιτούμενη επιλογή θα εκλεγεί εισάγοντας τον αντίστοιχο αριθμό επιλογής. Αν υπάρχουν λιγότερες από δέκα επιλογές μπορούμε να μεταχειριστούμε τη λειτουργία «GET» ώστε να μην χρειάζεται ο χειριστής να πατήσει μετά το RETURN. Οποιαδήποτε επιλογή κι αν έχει ενεργοποιηθεί, ο κατάλογος μπορεί να ξαναεμφανιστεί πατώντας ένα συγκεκριμένο πλήκτρο. Ένα πρόγραμμα κατευθυνόμενο από εντολές δεν έχει σελίδα καταλόγου επιλογών. Αντ' αυτού συνήθως υπάρχει μια γραμμή, ή καμμιά φορά δύο, κρατημένη στο κάτω μέρος όλων των οθονών γνωστή σα *γραμμή εντολών (command line)*. Εκεί συνήθως θα παρουσιάζεται μια υπενθύμιση όπως «Τι άλλο;» ακολουθούμενη από ένα δρομέα που αναβοσβήνει (flashing cursor). Η συγκεκριμένη επιλογή που απαιτείται εισάγεται με τη μορφή ενός μοναδικού γράμματος, ή ομάδας γραμμάτων, που διαλέγει ο προγραμματιστής και που έχει κάποια μνημονική συ-

σχέτισή της μ' αυτή την επιλογή. Ο χειριστής πρέπει να απομνημονεύσει τα γράμματα επιλογών αν και μια από τις επιλογές ίσως είναι η «Η» (που σημαίνει Help = Βοήθεια) η οποία εμφανίζει πλήρεις όλες τις δυνατότητες επιλογών.

Ένας έμπειρος χειριστής πιθανό να διάλεγε εύκολα ανάμεσα σε ένα πρόγραμμα κατευθυνόμενο από κατάλογο επιλογών και ένα από εντολές. Ένα πλεονέκτημα υπέρ του προγράμματος που κατευθύνεται από εντολές είναι η άνεση που κερδίζεται καθώς βλέπουμε την υπενθύμιση, την τρέχουσα επιλογή και τον δρομέα που αναβοσβήνει να παρουσιάζονται στο κάτω μέρος όλων των οθονών. Εν τούτοις, το πρόγραμμα το κατευθυνόμενο από κατάλογο επιλογών έχει σαφή πλεονεκτήματα. Πρώτον, δεν είναι ανάγκη να απομνημονεύουμε τις μνημονικές επιλογές καθώς η σελίδα καταλόγου εμφανίζεται σε μη-συντομογραφική μορφή. Δεύτερον, δεν χάνουμε καμμία γραμμή στο κάτω μέρος κάθε οθόνης. Τα προγράμματα σ' αυτό το βιβλίο θα είναι όλα κατευθυνόμενα από κατάλογο επιλογών. Εν τούτοις, δε θα είναι τόσο δύσκολο για οποιονδήποτε που θα ήθελε ένα σύστημα κατευθυνόμενο από εντολές, να μετατρέψει τις επιλογές της οθόνης σε εντολές.

Συχνά είναι βολικό να διασπάσουμε τον κατάλογο επιλογών σε δύο επίπεδα. Το πρώτο επίπεδο είναι για να κάνουμε μια *κύρια* επιλογή όπως «δημιουργία νέου αρχείου» ή «προσπέλαση σε υπάρχον αρχείο». Το δεύτερο επίπεδο είναι για να διαλέξουμε μία από τις πολλές διαθέσιμες επιλογές για την επεξεργασία ενός υπάρχοντος αρχείου. Τώρα θα περιγραφούν μερικές συνηθισμένες επιλογές επεξεργασίας. Θα πρέπει να τις καταλάβετε με τη γενική τους έννοια γιατί στην πράξη μπορεί να υπάρχουν διαφορές.

Δημιουργία νέου αρχείου

Η διαδικασία για τη δημιουργία ενός νέου αρχείου θα αποτελείται από μία σειρά υπενθυμίσεων, που καλούν το χειριστή να εισάγει πληροφορίες σχετικές με πράγματα όπως τα εξής:

- Όνομα αρχείου
- Μέγεθος αρχείου (μέγιστος αριθμός αναμενομένων εγγραφών)
- Αριθμός πεδίων
- Επικεφαλίδες πεδίων
- Εύρος πεδίων.
- Δεδομένα πεδίων (αριθμητικά ή αλφαριθμητικά).

Προσπέλαση σε υπάρχον αρχείο

Αν ένα αρχείο ήδη υπάρχει χρειαζόμαστε απλώς να διαλέξουμε μία από τις παρακάτω *δευτερεύουσες* επιλογές.

Καταχώρηση εγγραφής

Αυτή χρησιμοποιείται για:

(α) την καταχώρηση μιας πρόσθετης εγγραφής σε ένα αρχείο, που ήδη υπάρχει, ή

(β) την καταχώρηση της πρώτης εγγραφής σε ένα νέο αρχείο.

Παρουσίαση εγγραφής

Υπάρχουν αρκετοί τρόποι παρουσίασης πληροφοριών, που εξαρτιώνται σε μεγάλο βαθμό από τη μέθοδο που χρησιμοποιήθηκε κατά την οργάνωση του αρχείου. Μπορούν να ταξινομηθούν ως εξής:

(α) *Η παρουσίαση του πλατιού φύλλου*

΄Ολα τα πεδία κάθε εγγραφής υποτίθεται ότι βρίσκονται κατά μήκος μιας οριζόντιας γραμμής που απλώνεται πέρα από την ορατή περιοχή της οθόνης. Το πεδίο-κλειδί παραμένει *μόνιμα εμφανιζόμενο* στα αριστερά και τα άλλα πεδία παρουσιάζονται για να τα δούμε όσο χρειάζεται πατώντας αυθαίρετα επιλεγμένα πλήκτρα. Το πλεονέκτημα αυτής της παρουσίασης είναι η δυνατότητα να βλέπουμε πολλές εγγραφές ταυτόχρονα. Αν και την ίδια στιγμή παρουσιάζεται ένα τμήμα μόνο της κάθε εγγραφής μπορούμε να δούμε συγχρόνως τόσες εγγραφές, όσες είναι και οι διαθέσιμες γραμμές. ΄Αλλα αυθαίρετα επιλεγόμενα πλήκτρα χρησιμοποιούνται για να βρεθούμε σε άλλη ομάδα εγγραφών.

(β) *Η παρουσίαση μιάς εγγραφής*

Μία και μοναδική εγγραφή καταλαμβάνει την οθόνη, με κάθε πεδίο να αρχίζει σε νέα γραμμή. Αν η εγγραφή έχει πολύ μεγάλο μήκος για να εμφανιστεί ολόκληρη, συνήθως έχει κανονιστεί να παρουσιάζει κάθε φορά μία οθόνη πλήρη από δεδομένα.

Διαγραφή εγγραφής

Η δυνατότητα να διαγράφουμε μία εγγραφή είναι μία από εκείνες που θα δικαιολογούσαν τη χρήση ενός μηνύματος «Είσαι βέβαιος;». Μετά τη διαγραφή, θα παραμείνει ένα «κενό» στο αρχείο εκτός αν το πρόγραμμα περιέχει μία ρουτίνα που μετακινεί τις υπόλοιπες εγγραφές, καλύπτοντάς το. Μερικά προγράμματα θα παρεμβάλουν έναν ειδικό δείκτη, γνωστό σαν *ταφόπετρα (tombstone)*, στο χώρο που καταλάμβανε η διαγραμμένη εγγραφή. Η «ταφόπετρα» χρησιμοποιείται για να υποδείξει στο πρόγραμμα ότι η επόμενη εγγραφή που θα καταχωρηθεί μπορεί να τοποθετηθεί στο κενό αυτό χώρο.

Τροποποίηση εγγραφής

Οι εγγραφές είναι συχνά λανθασμένες. Μπορεί να καταχωρήθηκαν λανθασμένα ή να έγιναν αργότερα λανθασμένες λόγω αλλαγής των περιστάσεων. Κατά συνέπεια, όλα τα προγράμματα διαχείρισης αρχείων πρέπει να έχουν μία επιλογή διόρθωσης τμήματος, ή ολόκληρης της εγγραφής.

Αναζήτηση εγγραφής

΄Οπου είναι πρακτικό, ένα πρόγραμμα διαχείρισης αρχείων θα εξασφαλίζει ότι οποιαδήποτε εγγραφή η ομάδα εγγραφών που ικανοποιεί ένα ή περισσότερα κριτήρια αναζήτησης θα μπορεί να εντοπισθεί και να ανακτηθεί. Στ' αλήθεια, και μόνη της η λειτουργία της αναζήτησης είναι υπεραρκετή για να δικαιώσει τη χρήση του υπολογιστή για πληροφορίες διαχείρισης αρχείων. Η κανονική διαδικασία, που περιγράφτηκε παραπάνω, για την ανεύρεση μιας συγκεκριμένης εγγραφής θα καθορίσει το ειδικό πεδίο-κλειδί αναγνώρισης. Για παράδειγμα, το πεδίο-κλειδί σε ένα αρχείο των Πολιτικών όλου του Κόσμου θα είναι το όνομα του συγκεκριμένου πολιτικού. Μία αναζήτηση μέσω πεδίου-κλειδιού θα είναι πάντα η ταχύτερη μέθοδος. Εν τούτοις, μπορεί να υπάρχουν πολλοί λόγοι για τους οποίους μία εγγραφή δεν μπορεί να βρεθεί μ' αυτό τον τρόπο.

Για παράδειγμα, μπορεί να μην είσαστε τελείως σίγουροι για το όνομα του πολιτικού αλλά να έχετε την εντύπωση ότι στο όνομά του εμφανίζονται τα γράμματα «eag». Εδώ μπορεί να χρησιμοποιηθεί η επιλογή, «Αναζήτησε την υποαλφαριθμητική παράσταση». Για να βρούμε τον πολιτικό που μας διαφεύγει, η υποαλφαριθμητική παράσταση «eag» μπορεί να καταχωρηθεί κάτω από την επικεφαλίδα πεδίου «΄Ονομα». Αυτό θα μας παρουσιάσει εγγραφές όλων των πολιτικών που το όνομά τους περιέχει αυτή την υποαλφαριθμητική παράσταση, οπότε θα μπορούσαν ξαφνικά να εμφανιστούν στην οθόνη τα στοιχεία που έχουμε στο αρχείο για τον Reagan R.

Μία άλλη πιθανή ανάγκη αναζήτησης θα υπήρχε για εγγραφές στις οποίες οι πληροφορίες σε κάποιο από τα πεδία βρίσκονται μέσα σε κάποιο όριο ή όρια. Για παράδειγμα, σε ένα αρχείο για τροπικές ασθένειες, μπορεί να θέλαμε να βρούμε εκείνες που έχουν περίοδο επώασης στον άνθρωπο λιγότερο από 14 μέρες. Σε ένα άλλο αρχείο για διπολικά τρανζίστορ, μπορεί να θέλαμε να εξετάσουμε τις εγγραφές εκείνων με ανοδική τάση ρεύματος (Hfe) μεγαλύτερη από 100 αλλά μικρότερη από 130.

΄Ενας κατάλογος επιλογών αναζήτησης θα μπορούσε να εμφανιστεί ως εξής στη σελίδα καταλόγου:

1. Αναζήτησε τα πεδία με < >n
2. Αναζήτησε τα πεδία με = n
3. Αναζήτησε τα πεδία με > = n
4. Αναζήτησε τα πεδία με n
5. Αναζήτησε τα πεδία με > = n και < m
6. Αναζήτησε τα πεδία με υποαλφαριθμητική παράσταση.

Παράδειγμα 1
Αν ζητήσουμε την παραπάνω επιλογή 3 τότε το πρόγραμμα θα ζητούσε τις ακόλουθες πληροφορίες:

1. Επικεφαλίδα πεδίου; (Ας υποθέσουμε ότι απαντάμε Volts)
2. Δώσε αριθμό; (Ας υποθέσουμε ότι απαντάμε 3).

Τότε μετά την αναζήτηση θα παρουσιαστούν όλες οι εγγραφές με τιμή ίση ή μεγαλύτερη του 3 στη στήλη Volts

Παράδειγμα 2

Αν ζητήσουμε την επιλογή 5 παραπάνω και απαντήσουμε ως εξής:

1. Επικεφαλίδα πεδίου; (Ας υποθέσουμε ότι απαντάμε Volts).
2. Δώσε αριθμό; (Ας υποθέσουμε ότι απαντάμε 5).
3. Δώσε αριθμό; (Ας υποθέσουμε ότι απαντάμε 15).

Τότε μετά την αναζήτηση θα παρουσιαστούν όλες οι εγγραφές με τιμή στη στήλη Volts μεγαλύτερη ή ίση με 5 και μικρότερη από 15.

Τα δύο παραδείγματα αφορούσαν αναζήτηση αριθμητικών ορίων. Μερικά προγράμματα εργάζονται εξ ίσου καλά αν τα όρια δοθούν σε μορφή χαρακτήρων.

Παράδειγμα 3

Αν ζητήσουμε την επιλογή 4 και απαντήσουμε ως εξής:

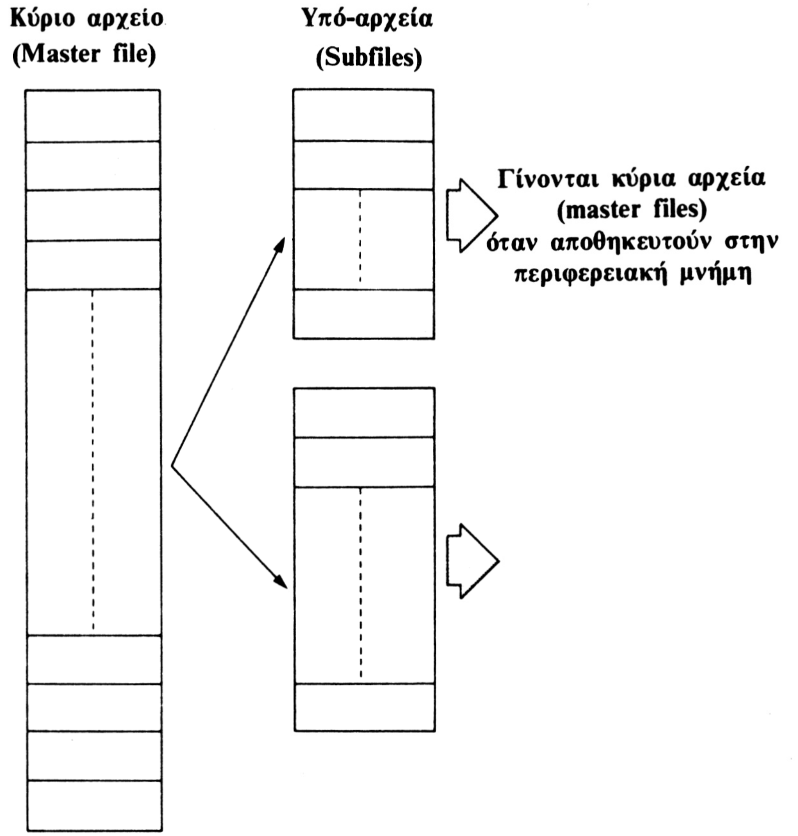
1. Επικεφαλίδα πεδίου; (Ας υποθέσουμε ότι απαντάμε όνομα)
2. Δώσε την αλφαριθμητική παράσταση; (Ας υποθέσουμε ότι απαντάμε «G»).

Μετά την αναζήτηση θα παρουσιαστούν όλες οι εγγραφές, στις οποίες το όνομα αρχίζει με γράμμα «προηγούμενο» του G.

Ταξινόμηση εγγραφών

Η δυνατότητα ταξινόμησης εγγραφών σε κάποια σειρά θεωρείται συχνά ουσιώδες στοιχείο κάθε καλού συστήματος διαχείρισης αρχείων. Ένα ταξινομημένο σύστημα έχει πάντα περισσότερη *πληροφοριακή αξία* από ένα μη ταξινομημένο σύστημα. Μία ειδική τεχνική, γνωστή σαν *δυαδική αναζήτηση (binary search)*, κάνει μία αναζήτηση να διεξάγεται πολύ γρηγορότερα από μία απλή γραμμική ή *σειριακή* αναζήτηση. Εν τούτοις, για να χρησιμοποιηθεί η δυαδική αναζήτηση, όλες οι εγγραφές πρέπει να είναι ταξινομημένες με βάση το πεδίο-κλειδί.

Είναι εύκολη η ταξινόμηση των εγγραφών αν ολόκληρο το αρχείο μπορεί πρώτα να φορτωθεί στη RAM. Εν τούτοις, η ταξινόμηση είναι μία σχετικά αργή διαδικασία στη BASIC, γι' αυτό, αν χρειάζεται μία επιλογή ταξινόμησης σε ένα μεγάλο αρχείο, είναι καλύτερο να προγραμματιστεί σε γλώσσα μηχανής.



Σχ. 2.2. Διάσπαση αρχείων.

Υπο-αρχεία (Subfiles)

Μερικά προγράμματα προσφέρουν τις διευκολύνσεις ενός «υπο-αρχείου», που περιέχει όλες τις εγγραφές που ικανοποιούν ένα ορισμένο κριτήριο αναζήτησης. Το υπο-αρχείο μπορεί τότε να αποθηκευτεί ξεχωριστά με ένα νέο όνομα αρχείου όπως φαίνεται στο Σχ. 2.2.

Αυτό μπορεί να χρησιμοποιηθεί συχνά για να διασπάσουμε ένα αρχείο που αρχίζει να δείχνει σημάδια υπερβολικής διόγκωσης. Η απάντηση στο ερώτημα πού να το διασπάσουμε μπορεί να δοθεί μετά από σκέψη αντί να κάνουμε έναν απλό αλφαβητικό διαχωρισμό. Ένα αρχείο για τα Βουνά της γης θα μπορούσε να διασπαστεί σε ξεχωριστά αρχεία σύμφωνα με την ήπειρο ή ίσως το ύψος πάνω από την επιφάνεια της θάλασσας. Αυτό θα είχε μεγαλύτερη πληροφοριακή αξία από ένα αρχείο διαχωρισμένο κατά τα ονόματα των βουνών σε «Α μέχρι G» και «Η μέχρι R», κ.λπ. Είναι γεγονός ότι ένα αρχείο που ξεκινάει τη ζωή του σαν ένα ανακάτωμα πληροφοριών δύσκολα

συνδεομένων μεταξύ τους, μπορεί, με τον προσεκτικό διαχωρισμό του σε υποαρχεία, να εξελιχθεί σε μία πηγή πληροφοριών με υψηλή οργάνωση. Αξίζει να σημειωθεί ότι όλες οι πληροφορίες είναι δεδομένα αλλά όλα τα δεδομένα δεν είναι πληροφορίες!

Επιλογές εκτύπωσης

Οι εκτυπωτές δεν είναι τυποποιημένοι. Αυτό καθιστά δύσκολο τον προγραμματισμό μιας επιλογής για την αποστολή επιλεγμένων δεδομένων αρχείων σε έναν εκτυπωτή. Οι πρώτοι εκτυπωτές ήταν απλή υπόθεση, κατάλληλοι μόνο για την εκτύπωση κεφαλαίων και μικρών χαρακτήρων σε μία επιφάνεια εκτύπωσης αλλά αυτή η εποχή τώρα έχει περάσει. Οι περισσότεροι σύγχρονοι εκτυπωτές έχουν τη δυνατότητα να ασχολούνται με πλάγια στοιχεία, υπογραμμίσεις, υπερυψωμένους και διπλού πάχους χαρακτήρες, με μεταβλητές αποστάσεις γραμμών και μερικοί επιτρέπουν ακόμη επιλογή χρώματος και κείμενο κατακορυφωμένο προς τα δεξιά. Δυστυχώς, οι χαρακτήρες ελέγχου για την ενεργοποίηση αυτών των πρόσθετων ιδιοτήτων ποικίλουν στους διάφορους εκτυπωτές. Βέβαια, αν χρησιμοποιηθεί ένας εκτυπωτής της Amstrad, δεν υπάρχει πρόβλημα με τους χαρακτήρες ελέγχου αλλά οι εκτυπωτές δεν είναι ακόμη αρκετά φτηνοί για να τους πετάμε κάθε φορά που αλλάζουμε υπολογιστή.

Οργάνωση αρχείου

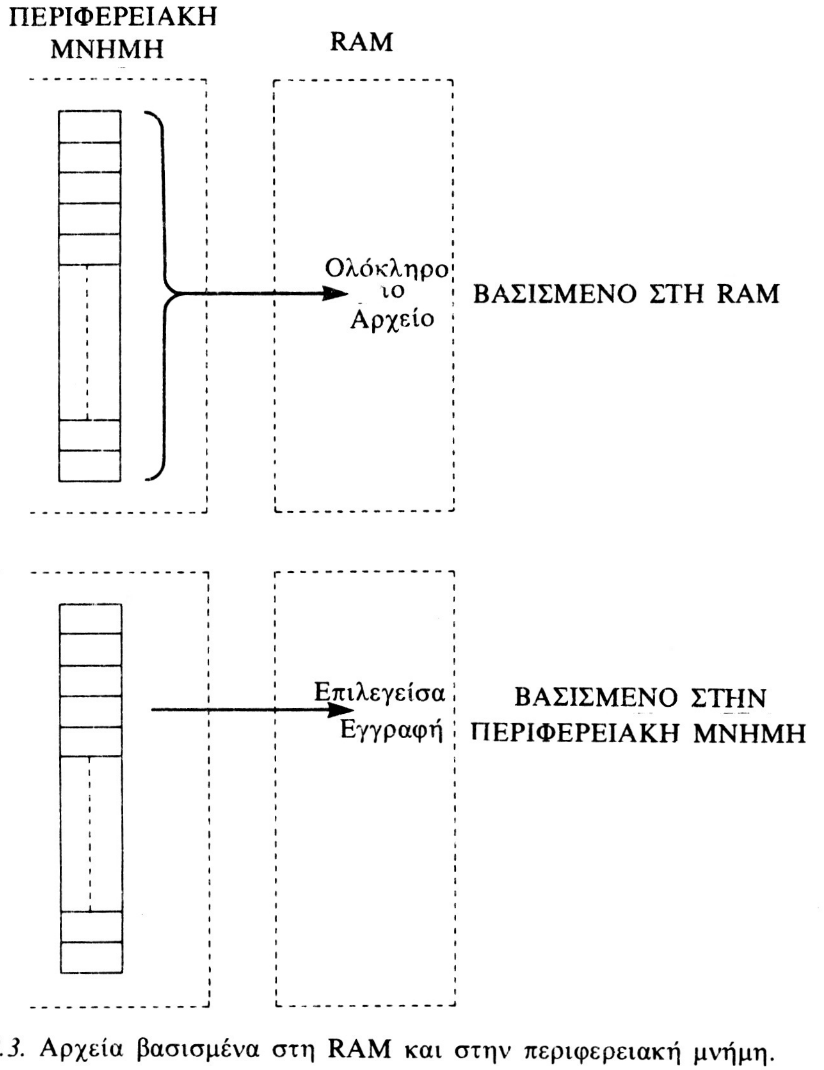
Ο κατάλογος των συνηθισμένων επιλογών επεξεργασίας αρχείου που περιγράψαμε παραπάνω έχει κύρια σημασία για εκείνους που απλώς θέλουν να χρησιμοποιήσουν το πρόγραμμα. Μπορεί να μη γνωρίζουν, ή πραγματικά δε χρειάζεται να το γνωρίζουν, πώς έχει προγραμματιστεί το αρχείο ή ποιά συγκεκριμένη *οργάνωση αρχείου* έχει ακολουθηθεί. Αν και δίνονται πλήρεις λίστες προγραμμάτων σ' αυτό το βιβλίο, ο σκοπός είναι να ενθαρρυνθούν οι αναγνώστες και να τις θεωρήσουν μόνο σαν οδηγό για να γράψουν τις δικές τους παραλλαγές. Η έμφαση από δω καί πέρα, θα μετατεθεί από το χρήστη προς τον προγραμματιστή.

Υπάρχουν αρκετοί, αναγνωρισμένοι τρόποι οργάνωσης ενός συστήματος διαχείρισης αρχείων. Το ποιά μέθοδος εκλέγεται εξαρτάται από:

(α) τον τύπο εξοπλισμού της περιφερειακής μνήμης.
(β) ποιές επιλογές θεωρούνται κύριας σημασίας.
(γ) το ποσό της χωρητικότητας περιφερειακής μνήμης που χρησιμοποιείται.

Λαμβάνοντας υπ' όψη το (α), η κασετοταινία πραγματικά δεν είναι χρήσιμη παρά μόνο για αρχεία βασισμένα στη RAM. Έτσι καθορίζομε τον ορισμό:

Ένα σύστημα διαχείρισης αρχείων βασισμένο στη RAM είναι εκείνο με το οποίο ολόκληρο το αρχείο φορτώνεται στη RAM πριν υπάρξει προσπέλαση ή επεξεργασία των εγγραφών.



Σχ. 2.3. Αρχεία βασισμένα στη RAM και στην περιφερειακή μνήμη.

Αν ένα αρχείο χρειάζεται ενημέρωση, πρέπει πάντα να φορτώνεται στη RAM, να γίνονται οι τροποποιήσεις και να ξαναγράφεται πάλι στην περιφερειακή μνήμη. Από την άλλη μεριά το *βασιζόμενο στην περιφερειακή μνήμη* ορίζεται ως εξής:

Ένα σύστημα διαχείρισης αρχείων βασισμένων στην περιφερειακή μνήμη είναι εκείνο στο οποίο ολόκληρο το αρχείο παρα-

μένει εκεί και μόνο η απαιτούμενη εγγραφή (ή μερικές φορές λίγες εγγραφές) μεταφέρεται στη RAM όταν απαιτείται.

Το Σχ. 2.3. δείχνει τις διαφορές μεταξύ τους. Το βιβλίο αυτό ασχολείται αποκλειστικά με αρχεία βασισμένα στη RAM που μπορούν να χρησιμοποιηθούν σε έναν υπολογιστή Amstrad.

Περίληψη

1. Ένα πρόγραμμα «φιλικό προς το χρήστη» δεν μπορεί ποτέ να είναι φιλικό για όλους τους χρήστες.
2. Αν κάποια εισαγωγή δεδομένων δεν είναι αποδεκτή, δεν σημαίνει ότι θα πρέπει να υπάρχει και το αντίστοιχο επεξηγηματικό μήνυμα.
3. Η απόκριση «είσαι βέβαιος» μπορεί να είναι χρήσιμη ή εκνευριστική. Εξαρτάται απ' το επίπεδο γνώσεων του χειριστή.
4. Θα πρέπει να είναι αδύνατο σ' έναν άπειρο χειριστή να μπορεί να «μπλοκάρει» το πρόγραμμα.
5. Η καλή υποστήριξη με γραπτή τεκμηρίωση αποτελεί σημαντική συμβολή στη φιλικότητα προς το χρήστη.
6. Ένα αρχείο δεδομένων δεν μπορεί να «σωθεί» (SAVE) ή να «φορτωθεί» (LOAD) άμεσα. Μπορούμε μόνο να έχουμε προσπέλαση σ' αυτό μέσα από κάποιο πρόγραμμα.
7. Ένα αρχείο αποτελείται από εγγραφές.
8. Μία εγγραφή αποτελείται από πεδία.
9. Εύρος πεδίου είναι ο αριθμός των χαρακτήρων του.
10. Η επικεφαλίδα πεδίου δίνει την έννοια που πρέπει να προσδοθεί στην πληροφορία μέσα στο πεδίο.
11. Αλφαριθμητικά πεδία είναι εκείνα που μπορούν να περιέχουν οποιοδήποτε χαρακτήρα (εκτός από χαρακτήρες ελέγχου).
12. Τα αριθμητικά πεδία περιέχουν μόνο αριθμούς, είτε ακέραιους είτε με κινητή υποδιαστολή.
13. Το πεδίο-κλειδί είναι εκείνο που μονοσήμαντα καθορίζει την εγγραφή.
14. Σε μεγάλα αρχεία προτιμούνται αριθμητικά πεδία-κλειδιά.
15. Το μέγεθος αρχείου αναφέρεται στον αριθμό των εγγραφών που περιέχει.
16. Ο χώρος που καταλαμβάνει ένα αρχείο στην περιφερειακή μνήμη εξαρτάται τόσο από το μέγεθος του όσο και από τα μεγέθη των πεδίων.
17. Αν ολόκληρο το αρχείο υφίσταται επεξεργασία μέσα στην μνήμη, λέγεται ότι είναι βασισμένο στη RAM. Το μέγεθος αρχείου περιορίζεται τότε από τη χωρητικότητα της RAM.
18. Αν μία συγκεκριμένη εγγραφή μπορεί να ληφθεί από την περιφερειακή μνήμη και να υποστεί επεξεργασία τότε λέγεται ότι το αρχείο είναι βασισμένο στην περιφερειακή μνήμη. Το μέγεθος αρχείου περιορίζε-

ται τότε από τη χωρητικότητα της περιφερειακής μνήμης.

19. Τα προγράμματα διαχείρισης αρχείων που κατευθύνονται από κατάλογο επιλογών περιστρέφονται γύρω από μία σελίδα επιλογών.

20. Στα προγράμματα διαχείρισης αρχείων, πού κατευθύνονται από εντολές, οι επιλογές διαλέγονται μέσω μιας «γραμμής - εντολής» (command line) στο κάτω μέρος όλων των οθονών.

21. Οι ρουτίνες ταξινόμησης είναι εύκολες αν το αρχείο είναι βασισμένο στη RAM.

22. Τα υποαρχεία περιέχουν επιλεγμένες εγγραφές μέσα σε βασικά αρχεία. Όταν αποθηκευτούν στην περιφερειακή μνήμη γίνονται τα ίδια κύρια αρχεία (master files).

23. Οι ρουτίνες εκτύπωσης μπορούν να εκμεταλλευθούν πλήρως τις δυνατότητες ενός εκτυπωτή μόνο αν είναι σκόπιμα σχεδιασμένες γι' αυτό το μοντέλο.

Ασκήσεις

2.1. Μπορεί ένα αρχείο δεδομένων να αποθηκευτεί χρησιμοποιώντας την εντολή SAVE «όνομα»;

2.2. Ποιο είναι το χαρακτηριστικό γνώρισμα του πεδίου-κλειδιού;

2.3. Κάτω από ποιές περιστάσεις είναι προτιμητέα τα αριθμητικά ή κωδικοποιημένα κλειδιά;

2.4. Αναφέρατε ένα σημαντικό πλεονέκτημα του αρχείου που βασίζεται στη RAM.

2.5. Ένα αρχείο που περιέχει 100 εγγραφές, καθεμιά με 5 πεδία ίσου μήκους, καταλαμβάνει περίπου 50K bytes μνήμης. Πόσοι χαρακτήρες υπάρχουν σε κάθε πεδίο, αγνοώντας γενικά τυχόν άλλους χαρακτήρες;

2.6. Σε μία παρουσίαση πλατιού φύλλου προς ποιά κατεύθυνση εμφανίζονται τα πεδία μιας εγγραφής;

2.7. Σε ποιές περιπτώσεις θα μπορούσε να χρησιμοποιηθεί η επιλογή «Αναζήτησε κάθε πεδίο με υποαλφαριθμητική παράσταση;»

2.8. Οι δυνατότητες διάσπασης ενός υπο-αρχείου είναι ιδιαίτερα χρήσιμες σε αρχεία βασισμένα στη RAM Γιατί;

Κεφάλαιο Τρία

Απλά Προγράμματα Διαχείρισης Αρχείων και Δομικά Στοιχεία

Το πλεονέκτημα των υπορουτινών

Τα προγράμματα σ' αυτό το βιβλίο προϋποθέτουν ότι ο αναγνώστης έχει ήδη κάποια πείρα στην BASIC του Amstrad, ιδιαίτερα στη χρήση υπορουτινών. Πιστεύεται μερικές φορές ότι οι υπορουτίνες θα 'πρεπε να χρησιμοποιούνται μόνο αν οι λειτουργίες που εκτελούν χρειάζονται περισσότερες από μία φορές στο ίδιο πρόγραμμα. Αν πάρουμε τη στενή άποψη της αποδοτικότητας υπό την έννοια του να πετύχουμε το μέγιστο αποτέλεσμα με τον ελάχιστο αριθμό γραμμών προγράμματος, μια τέτοια σκέψη είναι δικαιολογημένη. Εν τούτοις, η αποδοτικότητα μπορεί επίσης να σημαίνει μείωση του χρόνου προγραμματισμού, όχι μόνο στα αρχικά στάδια γραψίματος αλλά όταν απαιτούνται τροποποιήσεις ή προσθήκες καθώς αυξάνει η εμπειρία του χρήστη. Αν πάρουμε αυτή την ερμηνεία της αποδοτικότητας, τότε θα 'πρεπε να χρησιμοποιούνται υπορουτίνες σχεδόν για κάθε *λογικά διακεκριμένη* λειτουργία, ακόμη κι αν χρησιμοποιούνται στο πρόγραμμα μόνο για μία φορά. Μία υπορουτίνα είναι, ή θα 'πρεπε να είναι ένα σφιχτοδεμένο αυτάρκες «μαύρο κουτί» με *μία είσοδο* και *μία έξοδο*. Μια συλλογή υπορουτινών για γενικούς σκοπούς εισάγεται παρακάτω σ' αυτό το κεφάλαιο, και θα μας χρησιμεύσουν σα «δομικά στοιχεία» για ολοκληρωμένα συστήματα διαχείρισης αρχείων.

Αν υπάρχει οποιαδήποτε δυσκολία με την σύνταξη, θα πρέπει να συμβουλευτείτε κάποιο εγχειρίδιο της BASIC (σ.τ.μ. Π.χ. το Μάθετε Basic με τον Amstrad των εκδόσεων ΚΛΕΙΔΑΡΙΘΜΟΣ), γιατί για να εξοικονομήσουμε χώρο και να αποφύγουμε τις επαναλήψεις, θα πραγματευθούμε μόνο μερικές πλευρές της γλώσσας, που ενδιαφέρουν ειδικά τα αρχεία δεδομένων. Έχει καταβληθεί προσπάθεια να ακολουθήσουμε τα διδάγματα του δομημένου προγραμματισμού, τουλάχιστον όσο θα μας το επιτρέψει η BASIC. Δυστυχώς, το πολύ κακολογημένο GOTO δεν μπορεί να αποφευχθεί εντελώς. Ακόμη και μία υπορουτίνα είναι, στην πραγματικότητα, ένα GOTO. Εν τούτοις,

δεν είναι σοβαρό έγκλημα να χρησιμοποιούμε εντολές GOTO μέσα σε μία υπορουτίνα, υπό τον όρο ότι ο προορισμός του άλματος παραμένει μέσα στα όριά της. Το άλμα έξω από μία υπορουτίνα, πριν την κανονική της έξοδο, ακόμη κι αν έχει κανονιστεί να επιστρέφουμε πίσω πριν το κανονικό RETURN είναι μία παράβαση της όλης ιδέας του δομημένου προγραμματισμού. Αν μια υπορουτίνα έχει μόνο μία είσοδο και μόνο μία έξοδο είναι εύκολο να συνοδεύσει ένα πρόγραμμα και να κανονιστούν τροποποιήσεις χωρίς να δημιουργηθεί χάος.

'Ομορφες παρουσιάσεις στην οθόνη

Μπορούμε να ξεχάσουμε την ιδέα να εξωραΐσουμε τις παρουσιάσεις προσθέτοντας πλαίσια και διάφορα χρώματα, γιατί ένα πρόγραμμα με γοητευτική εμφάνιση είναι καλό αλλά σύνθετο, χωρίς αυτό να είναι αναγκαίο. Το κύριο αντικείμενο αυτού του βιβλίου είναι να σας βοηθήσει να καταλάβετε πώς δουλεύουν τα προγράμματα διαχείρισης αρχείων και πώς να τα προσαρμόσετε για να ταιριάζουν στους δικούς σας ειδικούς σκοπούς. Αυτό το αντικείμενο μπαίνει σε κίνδυνο αν τα κύρια γνωρίσματα ενός προγράμματος συσκοτίζονται από το στοίβαγμα γραφικών συμβόλων και χρωματικών πληροφοριών. Εν πάσει περιπτώσει είναι εύκολο να τα προσθέσετε όλα αυτά αργότερα.

Ονόματα μεταβλητών.

Οι μεταβλητές στις λίστες των προγραμμάτων σ' όλο το βιβλίο θα παίρνουν ονόματα με νόημα για να περικόψουμε τον αριθμό των REM (σχολίων). 'Οσο είναι δυνατό, τα ίδια ονόματα μεταβλητών θα έχουν ταυτόσημες λειτουργίες σε όλα τα προγράμματα. Οι επεξηγήσεις, που θα συνοδεύουν κάθε κυρίως πρόγραμμα, θα περιλαμβάνουν τις σημασίες των σπουδαιότερων ονομάτων μεταβλητών και τις λειτουργίες τους. Αυτό θα συνεπάγεται κάποια επανάληψη, αλλά καθιστά κάπως ευκολότερη την παρακολούθηση των προγραμμάτων.

Αρχεία βασισμένα σε πρόγραμμα

Στο Κεφάλαιο 2, αναφέραμε τους τρόπους οργάνωσης αρχείων που χρησιμοποιούνται συνήθως. Εν τούτοις, ένας απ' αυτούς παραλείφθηκε ηθελημένα γιατί είναι συζητήσιμο αν πρέπει ή όχι να θεωρηθεί στ' αλήθεια σύστημα διαχείρισης αρχείου. Ψάχνοντας για ένα καλύτερο όνομα, θα αναφερόμαστε σ' αυτό σαν αρχείο *βασισμένο στο πρόγραμμα (program - based)* γιατί το αρχείο κρατιέται σε εντολές τύπου DATA, και σε αριθμούς γραμμών, μέσα στο ίδιο το πρόγραμμα, αντί να αποθηκεύεται ξεχωριστά (θα μπορούσαμε να το έχουμε ονομάσει σύστημα «βασισμένο στα δεδομένα» (DATA - based) αλλά έτσι θα γινόταν σύγχυση με τη «βάση δεδομένων» (database)).

Επειδή οι πληροφορίες του αρχείου γράφονται μέσα στο πρόγραμμα, μόνο ένας «προγραμματιστής» μπορεί να το τροποποιήσει, έτσι η έκταση των επιλογών περιορίζεται αυστηρά. Πάντως μία τέτοια στοιχειώδης τακτοποίηση είναι θαυμάσια για να αποκτήσουμε μία προκαταρκτική ιδέα του θέματος και αξίζει να την εξετάσουμε.

Πρόγραμμα με εμφάνιση DATA

Είναι πάντα καλύτερο να αρχίζουμε από χαμηλά, έτσι ας μελετήσουμε τη λίστα του Προγράμματος 3.1, που όπως θα δείτε, είναι σίγουρα του χαμηλότερου επίπεδου. Το πρόγραμμα αυτό έχει έναν απλό σκοπό. Μόνο και μόνο παίρνει τις εγγραφές που γράφτηκαν σαν εντολές τύπου DATA και τις παρουσιάζει στην οθόνη. Παρ' όλα αυτά, σας ζητάμε να το πληκτρολογήσετε γιατί δείχνει πώς να παρουσιάζετε στην οθόνη τις πληροφορίες με πίνακα δύο διαστάσεων σε γραμμές και στήλες. Χρησιμοποιήστε το σαν «πειραματόζωο» τροποποιώντας μερικές από τις γραμμές και παρατηρώντας το αποτέλεσμα.

```
10 REM DISPLAY DATA STATEMENTS
20 READ filesize%,fields%
30 DIM A$(fields%,filesize%)
40 READ heading1$,heading2$
50 REM READ DATA INTO RECTANGULAR ARRAY
60 FOR R%=0 TO filesize%-1
70 FOR F%=0 TO fields%-1
80 READ A$(F%,R%)
90 NEXT:NEXT
100 GOSUB 150
110 END
120 '
130 '
140 REM VIEW FILE SUBROUTINE
150 CLS
160 PRINT TAB(7) "FILE CONTENTS"
170 PRINT "-----------------------------"
180 PRINT heading1$ TAB (19) heading2$
190 PRINT "-----------------------------"
200 FOR R%=0 TO filesize%-1
210 FOR F%=0 TO fields%-1
220 PRINT A$(F%,R%) TAB(19);
230 NEXT
240 PRINT
250 NEXT
```

```
260 RETURN
270 '
280 '
290 REM NUMBER OF RECORDS AND FIELDS
300 DATA 8,2
310 '
320 '
330 REM FIELD HEADINGS
340 DATA NAME,TELEPHONE
350 '
360 '
370 REM FILE INFORMATION
380 DATA DEREK,678 9993
390 DATA GILL,453 9345
400 DATA CATH,777 5847
410 DATA STEVE,345 6694
420 DATA JOHN,999 6836
430 DATA GRAYHAM,685 3737
440 DATA PAT,574 6858
450 DATA PAUL,786 8797
```

Πρόγραμμα 3.1. Εμφάνιση των εντολών τύπου DATA

Κατάλογος μεταβλητών χρησιμοποιούμενων στο Πρόγραμμα 3.1.

Filesize% = μέγεθος αρχείου = μέγιστος αριθμός εγγραφών σε κάθε αρχείο.
fields% = αριθμός πεδίων
heading1$ = επικεφαλίδα πεδίου 1
heading2$ = επικεφαλίδα πεδίου 2
F% = ένας δείκτης πεδίου
R% = ένας δείκτης εγγραφής
A$ (F%, R%) = ένα στοιχείο του πίνακα δύο διαστάσεων όπου κρατιούνται τα δεδομένα του κύριου αρχείου.

Απαιτούνται μερικές επεξηγήσεις όσον αφορά την A$ (F%, R%). Συνηθίζεται σε τέτοιους πίνακες αρχείου, να χρησιμοποιούμε τις μεταβλητές με την αντίθετη σειρά, την R% (αριθμός εγγραφής) πρώτα και F% (αριθμός πεδίου) ύστερα. Κανονικά, όταν προγραμματίζουμε εξ ολοκλήρου στη BASIC, πραγματικά δεν πειράζει αν ένα στοιχείο του πίνακα παίρνει το δείκτη A$ (R%, F%) ή A$ (F%, R%) εφ' όσον τηρείται συνέπεια. Εν τούτοις ένας πίνακας δύο διαστάσεων ή *ορθογωνικός* αποθηκεύεται στον Amstrad κατά στήλες και σε αύξουσα σειρά. Δηλαδή, η σειριακή σειρά αποθήκευσης αλφαριθμητικών δεικτών στη μνήμη είναι A$ (00) A$ (1,0) A$ (2,0) A$ (0,1) A$ (1,1) A$ (2,1)... A$ (0,η) A$ (1,η) A$(2,η). Συνεπώς, αν επιθυμούμε να εξα-

σφαλίσουμε σειριακή αποθήκευση των αλφαριθμητικών δεικτών πεδίων που έχουν σχέση με την κάθε μία εγγραφή, τότε προτιμάται η μορφή A$(F%, R%). Αυτή η οργάνωση αποθήκευσης ευνοεί γρήγορες και αποτελεσματικές μεθόδους ταξινόμησης ενός αρχείου σε πίνακα δύο διαστάσεων χρησιμοποιώντας γλώσσα μηχανής.

Οι εντολές τύπου DATA του Προγράμματος 3.1 αποτελούν τις πληροφορίες του αρχείου και έχουν μπει στο κάτω μέρος της λίστας. Επειδή βρίσκονται κάτω - κάτω, ξεχωρίζουν σαφώς αν, κάποια επόμενη φορά, χρειαστεί να τροποποιηθούν. Το αρχείο του παραδείγματος αποτελείται από οκτώ εγγραφές, δύο πεδίων η κάθε μία, που δίνουν το όνομα (NAME) και τον αριθμό τηλεφώνου (TELEPHONE) κάποιων φίλων μας.

Οι παράμετροι filesize% (μέγεθος αρχείου) και fields% (αριθμός πεδίων) που απαιτούνται από τους βρόχους καθώς και για την εντολή DIMension (διάσταση), διαβάζονται πρώτες. Τυχαίνει να είναι 8,2 (8 εγγραφές και 2 πεδία). Η εντολή DIMension δεν είναι αναγκαία στην BASIC του Amstrad για πίνακες με στοιχεία λιγότερα από δέκα αλλά πάντα υπάρχει η πιθανότητα να επιθυμήσετε να αυξήσετε τον αριθμό εγγραφών, έτσι είναι πάντα καλή τακτική να περιλαμβάνετε τις διαστάσεις (DIMensions).

Οι επικεφαλίδες κάθε πεδίου περιλαμβάνονται επίσης στις εντολές τύπου DATA και διαβάζονται στις μεταβλητές «heading1$» και «heading2$». Φυσικά, δεν υπήρχε πραγματική ανάγκη να σπαταλήσουμε αυτές τις δύο μεταβλητές γιατί οι επικεφαλίδες NAME και TELEPHONE θα μπορούσαν να είχαν εισαχθεί σαν σταθερές με μορφή κειμένου στο σώμα του προγράμματος, αλλά οι μεταβλητές τροποποιούνται ευκολότερα. Σε κάθε περίπτωση, στα προγράμματα που ακολουθούν, επιτρέπεται στο χρήστη να εισάγει οποιαδήποτε επικεφαλίδα χρειάζεται με την μορφή μεταβλητής.

Αφού ληφθούν οι αρχικές παράμετροι, οι εγγραφές μπορούν να διαβαστούν (READ) στα κατάλληλα στοιχεία A$(F%, R%) του πίνακα δύο διαστάσεων μέσω των δύο εμφωλιασμένων βρόχων FOR/NEXT (nested loops). Σημειώστε προσεκτικά ότι ο εξωτερικός βρόχος ελέγχει το R% και ο εσωτερικός βρόχος ελέγχει το F%.

Οι πληροφορίες που κρατιούνται σε γραμμές DATA είναι εύκολο να τροποποιηθούν χωρίς σοβαρό κίνδυνο για το υπόλοιπο πρόγραμμα, τόσο εύκολο, στ' αλήθεια, που σχεδόν δικαιολογεί τον τίτλο των «ημιμεταβλητών». Θα συμβουλεύαμε να διαθέτετε μία ξεχωριστή γραμμή για κάθε εγγραφή. Το στοίβαγμα μιας αλυσίδας εγγραφών σε μία γραμμή μπορεί να εξοικονομήσει λίγα bytes μνήμης αλλά μόνο με απώλεια της σαφήνειας.

Όπως αναφέρθηκε προηγούμενα, δεν υπάρχει σελίδα επιλογών και υπάρχει μόνο μία κύρια υπορουτίνα που εμφανίζει ολόκληρο το αρχείο με τον τρόπο του πλατιού φύλλου. Αυτό γίνεται από την υπορουτίνα VIEW FILE που απλώς ξετυλίγει τα δεδομένα του πίνακα A$(fields%, filesize%) και τα παρουσιάζει σε μορφή κατανοητή από ανθρώπους στην οθόνη. Οι επικεφαλίδες «heading1$» και «heading2$» περικλείονται σε εισαγωγικά - η μόνη μας παραχώρηση στη διακοσμητική. Για απλούστευση, και επειδή είναι το πρώτο πρόγραμμα, οι γραμμές σχεδιάζονται με απλές δηλώσεις PRINT, αλλά σε προγράμματα που θα ακολουθήσουν η σχεδίαση γραμμών θα ανατίθεται σε

ξεχωριστή υπορουτίνα. Προσέξτε ότι η γραμμή που εκτυπώνει (PRINT) τις εγγραφές τελειώνει με ένα «;» για να εμποδίσει μίαν αλλαγή γραμμής ώστε το πεδίο 2 (αριθμός τηλεφώνου) να εμφανιστεί στην ίδια γραμμή με το πεδίο 1 (όνομα).

Αν θέλετε να δοκιμάσετε με καινούρια δεδομένα, διαφορετικές επικεφαλίδες και περισσότερες εγγραφές, αντικαταστήστε απλώς τις γραμμές DATA εγγραφών αλλά θυμηθείτε επίσης να ενημερώσετε αντίστοιχα τις παραμέτρους των βρόχων και τις επικεφαλίδες πεδίων των δεδομένων. Αν αποφασίσετε να χρησιμοποιήσετε κι άλλα πεδία, η νέα διάταξη θα απαιτήσει τροποποίηση της υπορουτίνας VIEW. Στο παράδειγμα χρησιμοποιήθηκε το mode 1, ο τύπος των 40 χαρακτήρων ανά γραμμή, για λόγους αυξημένης καθαρότητας στην οθόνη. Όταν υπάρχουν μόνο δύο πεδία προς εμφάνιση, οι 40 στήλες είναι συνήθως υπεραρκετές, αν και το mode μπορεί εύκολα να αλλαχτεί για να διαθέτει 80 στήλες. Υπάρχουν αρκετά μικρά πράγματα που θα μπορούσαν να είχαν προστεθεί αλλά το πρόγραμμα είχε μόνο την πρόθεση να σπάσει τον πάγο - η φινέτσα αφήνεται για αργότερα.

Αρχείο βασισμένο σε πρόγραμμα με επιλογές

Η επόμενη λίστα, το Πρόγραμμα 3.2, αν και πρόκειται πάλι για ένα σύστημα διαχείρισης αρχείων βασισμένο στο πρόγραμμα, είναι κάπως λιγότερο πρωτόγονο γιατί είναι κατευθυνόμενο από κατάλογο Επιλογών (menu). Ομολογουμένως υπάρχουν μόνο τρεις επιλογές, και η μία απ' αυτές είναι EXIT PROGRAM (έξοδος από το Πρόγραμμα), αλλά θα βοηθήσει να αποκτήσουμε επίγνωση νέων πραγμάτων.

Ονόματα μεταβλητών

Για να διατηρήσουμε κάποια συνέχεια, τα προηγούμενα ονόματα μεταβλητών, έχουν κρατηθεί, αλλά, όπως δείχνει ο επόμενος κατάλογος, υπάρχουν ένα ή δύο ακόμα:

filesize% = μέγεθος αρχείου = μέγιστος αριθμός εγγραφών σε κάθε αρχείο
fields% = αριθμός πεδίων
heading1$ = επικεφαλίδα πεδίου 1
heading2$ = επικεφαλίδα πεδίου 2
F% = ένα ιδιαίτερο πεδίο
R% = μία ιδιαίτερη εγγραφή
A$(F%, R%) = ένα στοιχείο του δισδιάστατου πίνακα, όπου κρατιούνται τα δεδομένα του κύριου αρχείου
SEL% = αριθμός της εκλεγείσας επιλογής
K$ = μεταβλητή γενικής χρήσης
flag% = είναι ένας δείκτης αναζήτησης ναι/ όχι (1 = βρέθηκε εγγραφή, 0 = δεν υπάρχει εγγραφή).

```
10 REM SIMPLE PROGRAM BASED
20 REM FILING SYSTEM
30 READ filesize%,fields%
40 DIM A$(fields%,filesize%)
50 READ heading1$,heading2$
60 REM READ DATA INTO RECTANGULAR ARRAY
70 FOR R%=0 TO filesize%-1
80 FOR F%=0 TO fields%-1
90 READ A$(F%,R%)
100 NEXT:NEXT
110 '
120 '
130 GOSUB 210:REM MENU
140 IF SEL%=1 THEN GOSUB 390:GOTO 130
150 IF SEL%=2 THEN GOSUB 540:GOTO 130
160 CLS:PRINT "EXIT"
170 END
180 '
190 '
200 REM MENU SUBROUTINE
210 CLS
220 PRINT TAB(3) "PROGRAM BASED FILING S
YSTEM"
230 LOCATE 1,6
240 GOSUB 730:REM DRAW LINE
250 PRINT"(1) VIEW FILE"
260 PRINT"(2) DISPLAY RECORD"
270 PRINT"(3) EXIT PROGRAM"
280 GOSUB 730:REM DRAW LINE
290 LOCATE 1,15
300 PRINT"Select option ";
310 K$=INKEY$:IF K$="" THEN 310
320 SEL%=VAL(K$)
330 IF SEL%<1 OR SEL%>3 THEN 210
340 CLS
350 RETURN
360 '
370 '
380 REM VIEW FILE SUBROUTINE
390 PRINT TAB(7)"FILE CONTENTS"
400 GOSUB 730:REM DRAW LINE
410 PRINT heading1$ TAB(19) heading2$
420 GOSUB 730:REM DRAW LINE
430 FOR R%=0 TO filesize%-1
440 FOR F%=0 TO fields%-1
```

```
450 PRINT A$(F%,R%) TAB (18);
460 NEXT
470 PRINT
480 NEXT
490 GOSUB 780:REM HOLD DISPLAY
500 RETURN
510 '
520 '
530 REM GET RECORD SUBROUTINE
540 flag%=0
550 PRINT"ENTER "heading1$" REQUIRED ";
560 INPUT K$:K$=UPPER$(K$)
570 IF K$="" THEN 550
580 CLS:LOCATE 1,4
590 PRINT heading1$ TAB(19) heading2$
600 GOSUB 730:REM DRAW LINE
610 FOR R%=0 TO filesize%-1
620 FOR F%=0 TO fields%-1
630 IF A$(0,R%)=K$ THEN PRINT A$(F%,R%)
TAB(19);:flag%=1
640 NEXT
650 NEXT
660 IF flag%=0 THEN PRINT"THIS "heading1
$" NOT ON FILE"
665 PRINT
670 GOSUB 730:REM DRAW LINE
680 GOSUB 780:REM HOLD DISPLAY
690 RETURN
700 '
710 '
720 REM DRAW LINE SUBROUTINE
730 PRINT STRING$(40,CHR$(154))
740 RETURN
750 '
760 '
770 REM HOLD DISPLAY SUBROUTINE
780 PRINT:PRINT"PRESS ANY KEY TO CONTINU
19. E"
790 K$=INKEY$:IF K$="" THEN 790
20. 800 RETURN
810 '
820 '
21 830 REM NUMBER OF RECORDS AND FIELDS
840 DATA 8,2
2: 850 '
```

```
860 '
870 REM FIELD HEADINGS
880 DATA NAME,TELEPHONE
890 '
900 '
910 REM FILE INFORMATION
920 DATA DEREK,678 9993
930 DATA GILL,453 9345
940 DATA CATH,777 5847
950 DATA STEVE,345 6694
960 DATA JOHN,979 6836
970 DATA GRAYHAM,685 3737
980 DATA PAT,574 6858
990 DATA PAUL,786 8797
```

Πρόγραμμα 3.2. Αρχείο βασισμένο σε πρόγραμμα.

Το πρόγραμμα μας επιτρέπει να διαλέξουμε: είτε να δούμε ολόκληρο το αρχείο σε μορφή *πλατιού φύλλου* είτε να εμφανιστεί οποιαδήποτε εγγραφή επιλέξουμε. Μπορούμε να δούμε ότι το μεγαλύτερο τμήμα της δομής είναι όμοιο με το προηγούμενο πρόγραμμα. Για παράδειγμα, οι παράμετροι του βρόχου πληροφοριών αρχείου διαβάζονται ακόμα σε έναν πίνακα από εντολές τύπου DATA αλλά υπάρχουν μερικές πρόσθετες υπορουτίνες που χρειάζονται μελέτη.

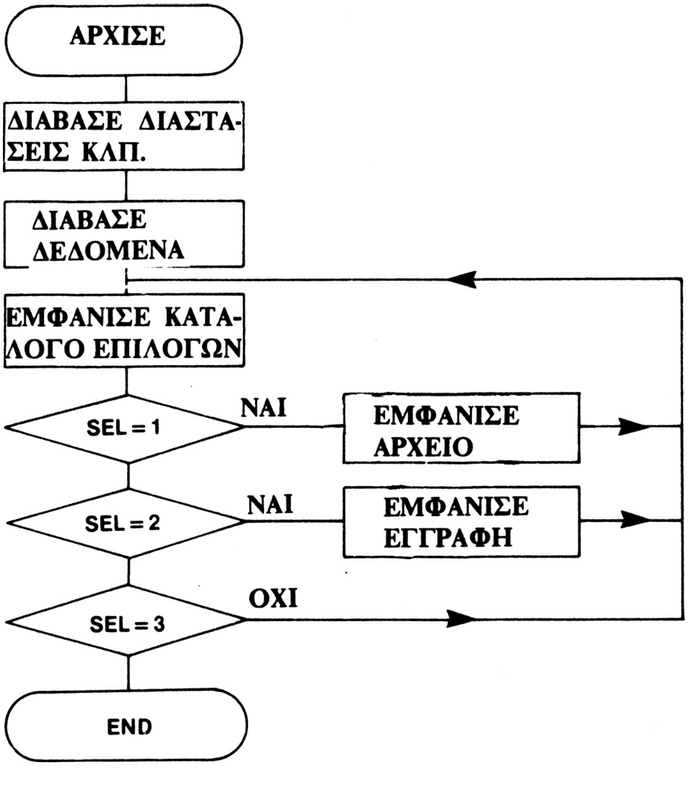
Υπορουτίνα MENU: εμφανίζει τον κατάλογο, βγάζει την υπενθύμιση, «SELECT OPTION» (διάλεξε επιλογή), και επιστρέφει με τον αριθμό εκλεγείσας επιλογής στη SEL%.
Υπορουτίνα VIEW FILE: αντίστοιχη της Επιλογής 1. Εκτυπώνει μια επικεφαλίδα και εμφανίζει τα δεδομένα αρχείου.
Υπορουτίνα GET RECORD: αντίστοιχη της Επιλογής 2. Επιτρέπει στο χρήστη να αναζητήσει μία συγκεκριμένη εγγραφή δίνοντας το *πεδίο-κλειδί*. Ο χρήστης καλείται να εισάγει το επιλεγμένο πεδίο-κλειδί με την υπενθύμιση που θα εμφανιστεί στην οθόνη σαν «ENTER NAME REQUIRED». Σημειώστε ότι «heading1$» είναι, στην περίπτωση αυτή, το NAME. Η απάντηση καταχωρείται στη μεταβλητή γενικής χρήσης K$ και μπαίνει και ένας έλεγχος για κενή αλφαριθμητική παράσταση (null string), αν ο χρήστης από απροσεξία πατήσει RETURN πριν δώσει το όνομα. Η αναζήτηση εκτελείται από το εξωτερικό βρόχο FOR/NEXT μετά την εμφάνιση των επικεφαλίδων. Καθώς διερευνώνται οι εγγραφές, το A$(0, R%), που είναι το πεδίο κλειδί όλων των εγγραφών, συγκρίνεται με το όνομα στην K$. Θυμηθείτε ότι R% είναι η μεταβλητή εγγραφής και 0 είναι το *πρώτο* πεδίο, που, εξ ορισμού, είναι το πεδίο-κλειδί της εγγραφής. Όταν βρεθεί το σωστό πεδίο, εκτυπώνεται ολόκληρη η εγγραφή καθώς ο εσωτερικός βρόχος FOR/NEXT περνάει και από τα δύο πεδία. Τα δεδομένα του παραδείγματος έχουν μόνο δύο πεδία αλλά η υπορουτίνα λειτουργεί για οποιοδήποτε αριθμό πεδίων. Αν βρεθεί πε-

δίο που να ταιριάζει, η flag% παίρνει την τιμή 1. Αν, μετά τη διερεύνηση ό-λων των εγγραφών, δεν βρεθεί πεδίο που να ταιριάζει, η flag% θα διατηρήσει την αρχική της τιμή 0 και θα παρουσιαστεί το μήνυμα «THIS NAME NOT ON FILE» (αυτό το όνομα δεν είναι στο αρχείο).

Υπορουτίνα DRAW LINE: δευτερεύουσα υπορουτίνα που χρησιμοποιείται για να σχεδιάσει μία στικτή γραμμή κατά μήκος της οθόνης.

Υπορουτίνα HOLD DISPLAY: δευτερεύουσα υπορουτίνα που σταματάει το πρόγραμμα μέχρι να πατηθεί οποιοδήποτε πλήκτρο, σε απάντηση κάποιας υπενθύμισης.

Το διάγραμμα ροής της Εικόνας 3.1 βοηθάει να παρακολουθήσουμε την παραπέρα δομή.



Σχ. 3.1. Διάγραμμα ροής για Πρόγραμμα 3.2.

Θα ήταν εύκολο να περιληφθεί μια επιλογή «αναζήτησης» για την εύρεση μιας εγγραφής μέσω του αριθμού τηλεφώνου (TELEPHONE) ως πεδίου-κλειδιού αλλά, όπως αναφέρθηκε νωρίτερα, αυτός ο τύπος οργάνωσης αρχείου δεν αξίζει να τον πάρουμε πολύ στα σοβαρά. Η αξία του έγκειται μόνο στην απλότητά του —ένα σκαλοπάτι για καλύτερα πράγματα.

Αποθήκευση και ανάκτηση αρχείων δεδομένων

Όλες οι πληροφορίες, που αποθηκεύονται σε κασετοταινίες ή δίσκους, λέγονται γενικά «αρχείο». Ακόμη κι όταν ένα πρόγραμμα σώζεται (IS SAVED) αναφέρεται συχνά σαν «αρχείο» αν και, για να το διακρίνουμε από *αρχεία δεδομένων* θα έπρεπε να λεγόταν *πρόγραμμα σε αρχείο*. Οι διαδικασίες διάσωσης και φόρτωσης προγραμμάτων έχουν γίνει πολύ εύκολες γιατί είναι λειτουργίες «ψωμοτύρι» σε κάθε οικιακό υπολογιστή. Θα ήταν ενοχλητικό και σίγουρα μη φιλικό προς το χρήστη να αναγκάσει το μέσο τελικό χρήστη να καταπιάνεται με το άνοιγμα και το κλείσιμο αρχείων κάθε φορά που ένα κανονικό πρόγραμμα χρειάζεται να σωθεί ή να φορτωθεί. Αντ' αυτού, οι περισσότερες συσκευές, όπως και ο CPC464, σώζουν και φορτώνουν προγράμματα μόλις λάβουν τις απλοποιημένες εντολές SAVE «όνομα» και LOAD «όνομα». Πολλά προγράμματα, ιδιαίτερα προγράμματα διαχείρισης αρχείων, απαιτούν τα δεδομένα τα χρησιμοποιούμενα από το πρόγραμμα, να αποθηκευτούν ξεχωριστά από το ίδιο το πρόγραμμα. Αυτές οι ξεχωριστές πληροφορίες αποθηκεύονται σαν αρχείο δεδομένων. Τα αρχεία δεδομένων δεν μπορούν να προσπελαθούν με τις κανονικές εντολές SAVE και LOAD γιατί αποθηκεύονται με τελείως διαφορετική διάταξη από εκείνη των αρχείων προγραμμάτων. Για παράδειγμα, όταν σώζεται ένα συνηθισμένο πρόγραμμα, οι λέξεις (keywords) της BASIC κωδικοποιούνται σε ομάδες των δύο bytes γνωστές σαν *λήμματα (tokens)*. Τα αρχεία δεδομένων, από την άλλη μεριά αποθηκεύονται κατ' ευθείαν σαν χαρακτήρες ASCII, με ειδικές πληροφορίες για τα άκρα τους και για τα άκρα των μπλοκ.

Άνοιγμα ενός αρχείου δεδομένων για έξοδο.

Στο σημείο του προγράμματος που πρέπει να αποθηκευτούν δεδομένα σε ταινία ή δίσκο, είναι ανάγκη να ανοίξουμε ένα αρχείο χρησιμοποιώντας την ακόλουθη δήλωση OPENOUT:

OPENOUT «όνομα αρχείου»

Το όνομα αρχείου μπορεί να κυριολεκτεί, όπως OPENOUT «Personnel» (προσωπικό), ή μία αλφαριθμητική μεταβλητή που έχει ήδη εκχωρηθεί, όπως OPENOUT Personnel$. Δεν επιτρέπονται πάνω από δεκαέξι χαρακτήρες για ένα όνομα αρχείου σε κασέτα ή οκτώ σε δίσκο.

Χρήση της PRINT#9

Άπαξ και το αρχείο έχει ανοιχτεί για έξοδο, τα δεδομένα μπορούν να γραφτούν στην ταινία μέσω μιας ειδικής εντολής εκτύπωσης που έχει τη διάταξη

PRINT # 9, κατάλογος μεταβλητών

Το« #9» πληροφορεί το σύστημα ότι η «εκτύπωση» πρέπει να εκτελεστεί στο *κανάλι εξόδου «9»*, που είναι το κασετόφωνο ή το σύστημα δίσκου, και όχι όπως συνήθως στην οθόνη του μόνιτορ. Ο όρος «κατάλογος μεταβλητών» σημαίνει μία ή περισσότερες μεταβλητές, χωριζόμενες από κόμματα, όπως

PRINT # 9, filesize%, fields%

'Οταν πρέπει να αποτυπωθεί σε ταινία ή δίσκο, ένα πλήρες αρχείο δεδομένων, είναι ανάγκη να αποθηκεύσουμε ορισμένα κύρια χαρακτηριστικά του αρχείου καθώς και τα κυριότερα στοιχεία της εγγραφής. Για παράδειγμα, όταν φορτώνεται το αρχείο, το πρόγραμμα που το φορτώνει δεν θα έχει καμμία πληροφορία για το μήκος του αρχείου ή τον αριθμό των πεδίων κάθε εγγραφής. Αυτές οι λεπτομέρειες πρέπει να αποκτηθούν απ' την ταινία δεδομένων πριν μπορέσουν να πάρουν τιμές οι παράμετροι βρόχων για την ανάγνωση του κυρίως όγκου του αρχείου.

Η ταχύτητα baud με την οποία στέλνονται τα δεδομένα στην ταινία θα είναι η αρχική ταχύτητα των 1000. Εν τούτοις, αν θέλετε να τα στείλετε με 2000 baud, τότε τη γραμμή OPENOUT, θα ακολουθήσει, η SPEEDWRITE 1.

Κλείσιμο αρχείου.

Αφού έχει γίνει η έξοδος των δεδομένων είναι σημαντικό να κλείσετε (CLOSE) το αρχείο όταν θα 'χετε τελειώσει μ' αυτό, γιατί μπορεί να υπάρχουν ακόμη δεδομένα στην περιοχή του buffer. Η ενέργεια του κλεισίματος του αρχείου διώχνει οποιαδήποτε δεδομένα παραμένουν στον buffer και τα γράφει στην ταινία ή στο δίσκο. Για να γίνει αυτό η αντίστοιχη λέξη (keyword) της BASIC είναι CLOSEOUT.

'Ανοιγμα ενός αρχείου για έξοδο

Για να ανακτήσουμε ένα αρχείο δεδομένων από ταινία ή δίσκο, είναι πάλι ανάγκη να ανοίξουμε ένα αρχείο, αλλά αυτή τη φορά, η λέξη (keyword) είναι OPENIN ακολουθούμενη από το όνομα του αρχείου.

OPENIN «όνομα - αρχείου» ή OPENIN αλφαριθμητική μεταβλητή

Χρήση του INPUT # *9*

Κανονικά η εντολή INPUT δέχεται δεδομένα από την κατ' εξοχήν περιφερειακή μονάδα εισόδου, που είναι το πληκτρολόγιο. Για να εισάγουμε

δεδομένα από ταινία ή δίσκο, η ισοδύναμη μορφή είναι:

INPUT #9, κατάλογος μεταβλητών

Για παράδειγμα,

INPUT #9, filesize%, fields%.

Αφού ανακτηθούν όλα τα δεδομένα, το αρχείο εισόδου πρέπει να κλειστεί κατά το συνηθισμένο τρόπο με την CLOSEIN.

Για να φανεί η διαδικασία γραψίματος και διαβάσματος δεδομένων, πληκτρολογήστε πρώτα τις λίγες γραμμές του προγράμματος 3.3.

```
10 REM STORING DATA ON TAPE
20 CLS
30 PRINT"ENTER SOME CHARACTERS "
40 LINE INPUT K$
50 OPENOUT "TEST"
60 PRINT#9,K$
70 CLOSEOUT
80 END
```

Πρόγραμμα 3.3. Αποθήκευση δεδομένων σε δίσκο ή σε ταινία.

Πριν τρέξετε το πρόγραμμα, βεβαιωθείτε ότι έχετε κατά μέρος μία διαθέσιμη κασέτα ξανατυλιγμένη ή μια δισκέτα για να δεχθεί τα δεδομένα. Μόλις τοποθετηθούν τα δεδομένα στην ταινία ή στο δίσκο κάντε ένα «hard reset» (ή βάλτε τον κομπιούτερ στο «off» και πάλι στο «on»). Τώρα πληκτρολογήστε το Πρόγραμμα 3.4 που ξαναδιαβάζει τα δεδομένα που αποθηκεύτηκαν μέσω του προηγούμενου προγράμματος.

```
10 REM READING DATA BACK FROM TAPE
20 CLS
30 PRINT"PLEASE REWIND DATA TAPE"
40 OPENIN "TEST"
50 INPUT#9,G$
60 CLOSEIN
70 PRINT"THE CHARACTERS READ BACK ARE"
80 PRINT G$
90 END
```

Πρόγραμμα 3.4. Διάβασμα δεδομένων από δίσκο ή από ταινία.

Αν έχετε ταινία πάλι, βεβαιωθείτε ότι η ταινία δεδομένων έχει ξανατυλιχτεί στο σωστό σημείο πριν τρέξετε το πρόγραμμα. Η συσκευή πρέπει να έχει αδειαστεί από δεδομένα με το «hard reset» μετά το Πρόγραμμα 3.3, οπότε τα δεδομένα που εκτυπώνονται από το πρόγραμμα 3.4 πρέπει να προέρχο-

νται από την ταινία ή τον δίσκο δεδομένων. Αν και η μεταβλητή K$ χρησιμοποιήθηκε για την εκτύπωση των δεδομένων στην ταινία και η G$ για την ανάκτησή τους, θα μπορούσε να είχε χρησιμοποιηθεί η ίδια μεταβλητή και στα δύο προγράμματα.

Δομικά στοιχεία γενικής χρήσεως

Το επόμενο κεφάλαιο είναι εξ ολοκλήρου αφιερωμένο στην περιγραφή ενός και μόνο πρακτικού συστήματος διαχείρισης αρχείων. Το πρώτο, και σχετικά μικρό κομμάτι του προγράμματος θα αναγνωρισθεί σαν ένα συνεκτικό *τμήμα ελεγχου (control section)*. Το υπόλοιπο πρόγραμμα είναι απλώς μια συλλογή από υπορουτίνες, που καλούνται, όταν χρειάζεται, από το τμήμα ελέγχου. Αν και μερικές υπορουτίνες είναι φτιαγμένες για να ταιριάζουν σ' αυτό το συγκεκριμένο πρόγραμμα, οι περισσότερες μπορούν να χρησιμοποιηθούν σαν τυπικά δομικά στοιχεία που μπορούν να συναρμοσθούν μέσα σε άλλα προγράμματα, είτε έτσι όπως είναι ή με ελαφρή τροποποίηση. Κατά την επινόηση δομικών στοιχείων προς χρήση από ένα φάσμα προγραμμάτων, υπάρχει πάντα το πρόβλημα του μεγέθους που θα ληφθεί υπ' όψη. Αν προσπαθήσουμε και βάλουμε πάρα πολλά τεχνάσματα μέσα σε μία υπορουτίνα, διακινδυνεύουμε το γενικό μας στόχο και παραβαίνουμε την κύρια κατευθυντήρια γραμμή που είναι ότι μία υπορουτίνα πρέπει να έχει μόνο μία σαφώς καθορισμένη λειτουργία. Όπως συμβαίνει μ' όλες τις κατευθυντήριες γραμμές, δεν είναι αναγκαία, και ούτε στ' αλήθεια συνιστάται η δουλική προσκόλληση σ' αυτή. Είναι συχνά ζήτημα προσωπικής κρίσης. Η συλλογή μιας βιβλιοθήκης δομικών στοιχείων μας γλυτώνει από το να 'χουμε να ξαναανακαλύψουμε τον τροχό σε κάθε καινούριο πρόγραμμα.

Τα ονόματα των μεταβλητών που χρησιμοποιούνται στα δομικά στοιχεία γενικής φύσης ίσως φαίνονται περίεργα όταν βρεθούν ξεχωριστά, αλλά η αλλαγή ονομάτων χρειάζεται μία επιπλέον εργασία. Ωστόσο, στα παραδείγματα που ακολουθούν, τα περισσότερα απ' τα ονόματα των μεταβλητών θα συμπίπτουν με αυτά που χρησιμοποιούνται στα κύρια προγράμματα που εμφανίζονται στα υπόλοιπα κεφάλαια αυτού του βιβλίου.

Δομικά στοιχεία υπορουτινών

Οι υπορουτίνες έχουν συχνά *ιεραρχημένη δομή*. Δηλαδή, μερικές υπορουτίνες καλούν άλλες υπορουτίνες να εκτελέσουν δευτερεύοντα καθήκοντα κατά τον ίδιο τρόπο που ένας μεγάλος οργανισμός αναθέτει μέρος της λιγότερο σημαντικής εργασίας σε υπεργολάβους. Είναι βολικό να κάνουμε διάκριση ανάμεσα σε *υψηλού* και *χαμηλού επιπέδου* υπορουτίνες. Για παράδειγμα, μπορεί να υπάρχει μία υπορουτίνα για να εμφανίζει μία από τις εγγραφές ενός αρχείου (μία υψηλού επιπέδου υπορουτίνα) που, με τη σειρά της, μπορεί να καλεί μία άλλη υπορουτίνα (μία υπορουτίνα χαμηλού επιπέδου) για τη

σχεδίαση γραμμών διαχωρισμού των πεδίων στην οθόνη. Η ενσωμάτωση της σχεδίασης γραμμών στην ίδια την υπορουτίνα εμφάνισης θα 'μοιαζε να είναι ο προφανής στόχος μας, αλλά αυτό δε θα ήταν πολύ αποδοτικό. Η σχεδίαση μιας γραμμής στην οθόνη είναι μία ασήμαντη, αλλά παρά ταύτα, χρήσιμη εργασία που πιθανότατα θα χρειαστεί σε πολλές άλλες υπορουτίνες. Αν και η εμφάνιση μιας εγγραφής είναι μια εργασία γενικής χρήσης σε οποιοδήποτε πρόγραμμα διαχείρισης αρχείων, η σχεδίαση μιας γραμμής είναι ακόμα γενικότερης χρήσης, έτσι ώστε να δικαιολογείται η κατάστρωση μιας ανεξάρτητης, χαμηλού επιπέδου υπορουτίνας.

Υπορουτίνα DRAWLINE

Καλείται με την GOSUB 2330.

```
2320 REM DRAW LINE
2330 PRINT STRING$(40,CHR$(154));
2340 RETURN
```

Η function STRING$ πρέπει να χρησιμοποιείται πάντα προτιμώντας την από την απλή μορφή PRINT «______________». Είναι κομψή και σίγουρα πιο οικονομική σε μνήμη. Ο CHR$ (154) είναι ένας ειδικός χαρακτήρας γραφικών (graphics character), σαράντα από τους οποίους χρησιμοποιούνται για να σχεδιάσουν τη γραμμή. Οποιοσδήποτε άλλος χαρακτήρας μπορεί να χρησιμοποιηθεί για να σχεδιαστεί η «γραμμή» αλλά αποφύγετε τον αστερίσκο — θυμίζει τις πρώτες μέρες των κομπιούτερ σπιτιού, όταν οι επιδεικτικές εμφανίσεις ήταν στην ημερήσια διάταξη.

Υπορουτίνα GET LINE INPUT

Καλείται με τη GOSUB 2260

```
2250 REM GET LINE INPUT
2260 LINE INPUT K$
2270 IF K$="" THEN 2260
2280 K$=UPPER$(K$)
2290 IF LEN(K$)>18 THEN K$=LEFT$(K$,18)
2300 RETURN
```

Όταν μία υπενθύμιση στην οθόνη ζητάει εισαγωγή αλφαριθμητικών παραστάσεων από το πληκτρολόγιο, μερικές αποκρίσεις δεν θα είναι αποδεκτές. Αυτή η υπορουτίνα απορρίπτει την άκυρη αλφαριθμητική παράσταση που προκαλεί ένας χειριστής που πάτησε κατά λάθος το πλήκτρο ENTER πριν δώσει κάποια λέξη. Η χρήση της LINE INPUT συνιστάται σε προγράμματα διαχείρισης αρχείων γιατί το κείμενο συχνά περιέχει σημεία στίξης και άλλους χαρακτήρες που η κανονική δήλωση INPUT θα απέρριπτε. Το κείμενο καταχωρείται σε μία προσωρινή *σφαιρική (global)* μεταβλητή K$. Αφού η υπορουτίνα επιστρέφει, το κείμενο στην K$ θα καταχωρηθεί κανονι-

κά σε μία άλλη μεταβλητή. Η υπορουτίνα δίνει επίσης στο χρήστη την ελευθερία να εισάγει το κείμενο είτε σε κεφαλαία είτε σε μικρά γράμματα. Έχει χρησιμοποιηθεί η υπέροχη μικρή λέξη (keyword) UPPER$ του Amstrad, που εξασφαλίζει ότι όλο το κείμενο που εισαγάγαμε, ανεξάρτητα αν καταχωρήθηκε με κεφαλαία ή μικρά γράμματα, θα μετατραπεί σε κεφαλαία. Είναι σημαντικό, όταν γίνεται ταξινόμηση αλφαριθμητικού κειμένου, οι χαρακτήρες να είναι ή όλοι σε κεφαλαία ή όλοι σε μικρά γιατί η ταξινόμηση πραγματοποιείται βάσει των τιμών του ASCII κώδικα. Αν δεν ακολουθηθούν κάποια βήματα για την επανόρθωση ενός κειμένου μικτής μορφής (με κεφαλαία και μικρά), η ταξινόμηση θα μπορούσε να δώσει αποτελέσματα αλλ' αντ' άλλων.

Η υπορουτίνα περιορίζει επίσης το μήκος όλου του εισαγόμενου στοιχείου σε 18 χαρακτήρες. Αυτό το υλοποιεί αποκόβοντας τους χαρακτήρες που υπερβαίνουν τους 18 και όχι απορρίπτοντας ολόκληρο το εισαγόμενο στοιχείο. Αυτός είναι ένας περιορισμός, ειδικής μάλλον παρά γενικής χρήσης, και η αντίστοιχη γραμμή μπορεί πάντα να τροποποιηθεί ή να αφεθεί τελείως απ' έξω. Εδώ έχει ενσωματωθεί μόνο και μόνο επειδή ικανοποιεί τις ανάγκες του Προγράμματος 4.1 που παρουσιάζεται στο επόμενο κεφάλαιο.

Υπορουτίνα PRESS ANY KEY

Καλείται με τη GOSUB 2530

```
2510 REM PRESS ANY KEY
2520 PRINT:PRINT"Press any key to contin
ue"
2530 K$=INKEY$:IF K$="" THEN 2530
2540 RETURN
```

Αυτή θέτει σε λειτουργία τον ευρέως γνωστό τρόπο σταθεροποίησης μιας οθόνης ή το σταμάτημα ενός προγράμματος μέχρι να ετοιμαστεί ο χειριστής. Παρατηρήστε πάλι την εμφάνιση της πανταχού παρούσας μεταβλητής K$. Οπουδήποτε στα προγράμματά μας παρουσιάζεται η K$ μπορεί να θεωρηθεί σαν προσωρινή μεταβλητή. Η χρήση της ίδιας πάντα σφαιρικής μεταβλητής είναι καλή τακτική. Εξοικονομεί μνήμη και απλοποιεί την παρουσίαση ενός προγράμματος. Πάρα πολλές διαφορετικές μεταβλητές σε μία λίστα ενός προγράμματος, κάνουν δύσκολη την παρακολούθησή του.

Υπορουτίνα GET FILENAME

Καλείται με τη GOSUB 2370.

```
2360 REM GET FILENAME
2370 CLS:PRINT"Enter filename"
2380 GOSUB 2260
2390 IF LEN(K$)>16 THEN PRINT "Too long:
Enter again":GOTO 2380
```

```
2400 filename$=K$
2410 RETURN
```

Το όνομα αρχείου απαιτείται σε αρκετά σημεία σ' ένα σύστημα διαχείρισης αρχείων. Η υπενθύμιση για το όνομα εμφανίζεται στην οθόνη και η απάντηση λαμβάνεται και ελέγχεται από την υπορουτίνα GET LINE INPUT. Η πρόσθετη μείωση σε 16 γραμμές είναι για να ικανοποιήσει τον περιορισμό 16 χαρακτήρων για ονόματα αρχείων. Παρατηρήστε ότι η μεταβλητή, filename$ καταχωρείται στην K$ πριν την τελική επιστροφή.

Υπορουτίνα SAVE FILE

Καλείται με τη GOSUB 700.

```
690 REM SAVE FILE
700 GOSUB 2370
710 SPEED WRITE 1
720 OPENOUT filename$
730 PRINT#9,filesize%,fields%,L%
740 FOR R%=0 TO L%
750 FOR F%=0 TO fields%
760 PRINT#9,A$(F%,R%)
770 NEXT:NEXT
780 CLOSEOUT
790 RETURN
```

'Εχει ήδη δοθεί η πληροφόρηση σε γενικές γραμμές για το άνοιγμα και το κλείσιμο αρχείων. Αυτή είναι μία πλήρης υπορουτίνα για την εκτύπωση ενός αρχείου δεδομένου σε μία ταινία η δίσκο. Προϋποθέτει ότι ορισμένες πληροφορίες βρίσκονται στις παρακάτω μεταβλητές πριν κληθεί:

Μέγιστος επιτρεπόμενος αριθμός εγγραφών στη filesize%
Μέγιστος επιτρεπόμενος αριθμός πεδίων στη fields%
Τρέχων αριθμός εγγραφών στην L%
'Ολο το κείμενο στον πίνακα, A$(F%, R%).

Η υπορουτίνα καλεί πρώτα την GET FILE NAME για να ανοίξει το αρχείο για έξοδο. Τρία στοιχεία, που αποτελούν τις επί κεφαλής πληροφορίες, στέλνονται πρώτα στην ταινία ή στο δίσκο. Τα δύο πρώτα χρειάζονται για να τροφοδοτήσουν την εντολή DIMension όταν το αρχείο δεδομένων ξαναδιαβαστεί αργότερα. Το τρίτο στοιχείο, L%, χρειάζεται σαν μία παράμετρος του βρόχου FOR/NEXT. Οι δύο εμφωλιασμένοι βρόχοι εκτυπώνουν τον πίνακα στην ταινία ή στο δίσκο. Ο εσωτερικός βρόχος αναλαμβάνει τα πεδία κάθε εγγραφής και ο εξωτερικός βρόχος αναλαμβάνει ολόκληρες τις εγγραφές. Δεν χρειάζονται άλλες υπενθυμίσεις στην οθόνη για τη λειτουργία του κασετόφωνου ή του δίσκου γιατί βγαίνουν αυτόματα με την εντολή OPEN.

Υπορουτίνα LOAD FILE

Καλείται με τη GOSUB 570.

```
560 REM LOAD FILE
570 GOSUB 2370
580 OPENIN filename$
590 INPUT#9,filesize%,fields%,mainsize%
600 flag%=1
610 DIM A$(fields%,filesize%)
620 FOR R%=0 TO mainsize%
630 FOR F%=0 TO fields%
640 INPUT#9,A$(F%,R%)
650 NEXT:NEXT
660 CLOSEIN
670 RETURN
```

Μετά από μία προκαταρκτική κλήση για το όνομα αρχείου, διαβάζονται τα τρία στοιχεία των επικεφαλής πληροφοριών. Οι πρώτες δύο μεταβλητές είναι ταυτόσημες με εκείνες που χρησιμοποιήθηκαν στην υπορουτίνα SAVE αλλά η τρίτη, αν και είχε κρατηθεί με το όνομα L%, αλλάζεται σε mainsize% (βασικό μέγεθος). Ο λόγος αυτής της αλλαγής μπορεί προς στιγμή να μη φαίνεται, αλλά έχει να κάνει με προγράμματα που μπορεί να διακρίνουν μεταξύ *βασικού αρχείου (main file)* και *υποαρχείου (subfile)*. Οι εμφωλιασμένοι βρόχοι FOR/NEXT για το διάβασμα του πίνακα είναι της ίδιας μορφής με εκείνους της υπορουτίνας SAVE, εκτός, βέβαια, από την αντικατάσταση της PRINT#9 από την INPUT#9.

Υπορουτίνα DISPLAY FILE

Καλείται με τη GOSUB 820.

```
810 REM DISPLAY FILE
820 F%=1:top%=1
830 WHILE INKEY(47)<>0
840 CLS:PRINT"Press space bar to regain
menu"
850 GOSUB 2330
860 PRINT A$(0,0) TAB(20) A$(F%,0)
870 GOSUB 2330:bottom%=top%+19
880 IF bottom%>L% THEN bottom%=L%
890 FOR R%=top% TO bottom%
900 PRINT A$(0,R%) TAB(20) A$(F%,R%)
910 NEXT
920 CALL &BB03:'KM RESET
```

```
930 CALL &BB18:'KM WAIT KEY
940 IF INKEY(8)=0 THEN F%=F%-1
950 IF INKEY(1)=0 THEN F%=F%-1
960 IF INKEY(0)=0 THEN top%=top%-20
970 IF INKEY(2)=0 THEN top%=top%+20
980 IF F%<1 THEN F%=fields%
990 IF F%>fields% THEN F%=1
1000 IF top%<1 THEN top%=(INT(L%/20)*20)
+1
1010 IF top%>L% THEN top%=1
1020 WEND
1030 RETURN
```

Αυτή η υπορουτίνα εμφανίζει στην οθόνη σελίδες του αρχείου σε μορφή *πλατιού φύλλου,* δηλαδή μία σελίδα γεμάτη εγγραφές), αλλά λόγω του περιορισμένου πλάτους της οθόνης, μόνο το πεδίο-κλειδί και ένα ακόμη πεδίο μπορούν να φαίνονται συγχρόνως. Το πεδίο-κλειδί φαίνεται πάντα ενώ τα άλλα πεδία μπορούν να έρθουν «πλάϊ» σ' αυτό χρησιμοποιώντας τα πλήκτρα δεξιάς ή αριστερής μετατόπισης. Οι άλλες σελίδες εγγραφών μπορούν να εμφανιστούν μέσω των πλήκτρων άνω και κάτω μετατόπισης. Η μάλλον μυστηριώδης CALL & BB03 είναι μία ρουτίνα του λειτουργικού συστήματος σε γλώσσα μηχανής, που, μεταξύ άλλων καθαρίζει (επαναφέρει στην αρχική του κατάσταση) τον buffer του πληκτρολογίου. Η άλλη κλήση σε γλώσσα μηχανής της & BB18 είναι η KM WAIT KEY (περίμενε για το επόμενο πλήκτρο). Τα γράμματα «KM» στα προσαρτημένα σχόλια (REM) αναφέρονται στον «Keyboard Manager» (Διαχειριστή Πληκτρολογίου) και είναι το ομαδικό όνομα που δόθηκε από τους σχεδιαστές του συστήματος ROM σε ορισμένες από τις ρουτίνες του λειτουργικού τους συστήματος.

Κατά συνθήκη, οι επικεφαλίδες των πεδίων —για παράδειγμα οι NAME, AGE, TRADE κ.λπ.— αποθηκεύονται στα στοιχεία «Record 0» (Εγγραφή 0) του βασικού πίνακα. Οι αριθμοί πεδίου-κλειδιού είναι από 0 μέχρι η, ενώ το πεδίο-κλειδί είναι πάντα στο πεδίο 0. Αυτό μπορεί να εξηγήσει γιατί η γραμμή 860 εμφανίζει τις επικεφαλίδες πεδίων στο πάνω μέρος της οθόνης. Η A$ (0,0) είναι η επικεφαλίδα πεδίου-κλειδιού και A$ (F%, 0) είναι η επικεφαλίδα του πεδίου F%. Οι γραμμές 890 μέχρι 910 εμφανίζουν ταυτόχρονα δύο πεδία δεδομένων το πολύ 20 εγγραφών. Σημειώστε ότι το κύριο μέρος της υπορουτίνας βρίσκεται μέσα στα όρια ενός βρόχου WHILE/WEND που μας ζητάει να πατήσουμε ένα πλήκτρο κενού διαστήματος (47). Οι τέσσερις άλλες γραμμές INKEY που βρίσκονται παρακάτω ζητάνε να πατήσουμε ένα πλήκτρο του δρομέα (cursor key).

Υπορουτίνα FIND RECORD

Καλείται με τη GOSUB 2670.

```
2660 REM FIND RECORD
2670 R%=0
2680 PRINT"Give record entry under ";A$(
F%,0)
2690 GOSUB 2260
2700 R%=R%+1
2710 IF K$=LEFT$(A$(F%,R%),LEN(K$)) THEN
 2740
2720 IF R%<L% THEN 2700
2730 PRINT" No such record on file":GOTO
 2670
2740 RETURN
```

Μία αναζήτηση διεξάγεται μέσα στο αρχείο για μία συγκεκριμένη εγγραφή μέσω μιας από τις επικεφαλίδες πεδίων, έτσι πρέπει να περαστεί ο αριθμός πεδίου F%. Αν η αναζήτηση γίνεται μέσω του πεδίου-κλειδιού, τότε το F% θα περαστεί σαν 0. Αν υποθέσουμε ότι η επικεφαλίδα του πεδίου-κλειδιού είναι NAME, η πρώτη υπενθύμιση θα λέει:

> «Give record entry under NAME» (= Δώσε την προς είσοδο εγγραφή μέσω του NAME).

Η απάντηση που δίνουμε από το πληκτρολόγιο καταχωρείται στην K$, με κλήση της GET LINE INPUT. Η γραμμή 2710 προσδίνει ένα βαθμό φιλικότητας προς το χρήστη στην απάντηση από το πληκτρολόγιο. Το πρώτο γράμμα, ή τα λίγα πρώτα γράμματα, του ονόματος που καταχωρήθηκε στην K$ αρκούν για την αναζήτηση. Για παράδειγμα, αν η εγγραφή μέσω ονόματος ήταν BREW P, τα γράμματα BR θα αρκούσαν ίσως για να ανακτήσουμε την εγγραφή. Η αναζήτηση θα αποφέρει την πρώτη εγγραφή που έχει τα BR σαν τους δύο πρώτους χαρακτήρες μέσα στο όνομα. Υπάρχει, βέβια η πιθανότητα να βρίσκεται επίσης στο αρχείο ο BROWN GH, περίπτωση κατά την οποία η αναζήτηση θα πρέπει να ξαναγίνει, χρησιμοποιώντας τα τρία πρώτα γράμματα. Το τελικό αποτέλεσμα της υπορουτίνας είναι η παραμονή του αριθμού της εγγραφής στην R%. Η αναζήτηση εκτελείται μέχρι να βρεθεί η εγγραφή ή να διαπιστωθεί ότι δεν βρίσκεται στο αρχείο.

Υπορουτίνα DISPLAY RECORD

Καλείται με τη GOSUB 2040.

```
2030 REM DISPLAY RECORD
2040 F%=0:GOSUB 2670:CLS
2050 PRINT:PRINT"Current Record number"R
%:PRINT
2060 GOSUB 2330
2070 FOR F%=0 TO fields%
```

```
2080 PRINT A$(F%,0) TAB(21) A$(F%,R%)
2090 NEXT
2100 GOSUB 2330
2110 GOSUB 2520
2120 RETURN
```

Η υπορουτίνα καλεί πρώτα τη FIND RECORD για να έχει τον αριθμό της εγγραφής στην R%. Τότε εμφανίζεται στην οθόνη ολόκληρη η εγγραφή με επικεφαλίδα της τον τρέχοντα αριθμό εγγραφής. Ο όρος «τρέχων αριθμός» είναι σημαντικός γιατί η σταθερότητα των αριθμών εγγραφών δεν είναι εγγυημένη. Αν κάποτε διαγραφεί μία εγγραφή από το αρχείο, οι αριθμοί εγγραφών ανακατανέμονται για να καλύψουν το χάσμα που αλλιώς θα παρουσιαζόταν. Οι δύο κλήσεις στο κάτω μέρος της υπορουτίνας είναι οι DRAW LINE και PRESS ANY KEY.

Υπορουτίνα FIND FIELD HEADING

Καλείται με τη GOSUB 2570.

```
2560 REM FIND FIELD HEADING
2570 F%=-1
2580 PRINT"Operate on which field? (Give
 heading)
2590 GOSUB 2260
2600 F%=F%+1
2610 IF K$=LEFT$(A$(F%,0),LEN(K$)) THEN
2640
2620 IF F%<fields% THEN 2600
2630 PRINT"No such field":GOTO 2570
2640 RETURN
```

Κανονικά, μία αναζήτηση στο αρχείο καθοδηγείται μέσω της επικεφαλίδας του πεδίου-κλειδιού αλλά αυτή η υπορουτίνα χρειάζεται για να υποστηρίζει αναζητήσεις μέσω της επικεφαλίδας οποιουδήποτε πεδίου. Η απάντηση στην υπενθύμιση θα είναι στην K$ μετά από μία κλήση της GET LINE INPUT. Η γραμμή 2610, που τώρα θα πρέπει να σας είναι οικεία, επιτρέπει στο χειριστή, να επιλέξει να δώσει το πρώτο ή τα δύο πρώτα γράμματα ή ολόκληρη την επικεφαλίδα.

Υπορουτίνα PRINT FILE

Καλείται με τη GOSUB 1940.

```
1930 REM PRINT FILE
1940 FOR R%=1 TO L%
1950 PRINT#8:PRINT#8,"Record No :"R%
1960 PRINT#8,STRING$(40,"-")
```

```
1970 FOR F%=0 TO fields%
1980 PRINT#8,A$(F%,0) TAB(21) A$(F%,R%)
1990 NEXT
2000 NEXT
2010 RETURN
```

Ανάλογα με τις ιδιότητες του χρησιμοποιούμενου τύπου εκτυπωτή, αυτή η υπορουτίνα εκτυπώνει ολόκληρο το αρχείο στο χαρτί, περιλαμβανομένων και των τρεχόντων αριθμών εγγραφών. Δεν έχει ληφθεί καμμία πρόνοια για την αποστολή χαρακτήρων ελέγχου στον εκτυπωτή γιατί δεν είναι τυποποιημένοι αλλά, αν χρειαστεί, μπορούν να δοθούν σε μία νέα γραμμή αμέσως κάτω από την εντολή REM.

Υπορουτίνα TOTALISE COLUMN

Καλείται με τη GOSUB 2150.

```
2140 REM TOTALISE COLUMN
2150 total=0
2160 GOSUB 2570
2170 FOR R%=1 TO L%
2180 total=total+VAL(A$(F%,R%))
2190 NEXT
2200 PRINT"Column total ="total
2210 PRINT"Column average="total/L%
2220 GOSUB 2520
2230 RETURN
```

Η επικεφαλίδα της στήλης που πρόκειται να αθροιστεί λαμβάνεται καλώντας την FIND FIELD HEADING. Εκτυπώνονται τόσο το άθροισμα της στήλης όσο και ο μέσος όρος της στήλης. Επειδή σε μία αλφαριθμητική μεταβλητή οι αριθμητικές πράξεις εκτελούνται με τη χρήση της VAL, το αποτέλεσμα θα είναι χωρίς κανένα νόημα αν η εκλεγείσα στήλη περιέχει αλφαβητικούς χαρακτήρες

Υπορουτίνα SORT FILE

Καλείται με τη GOSUB 1490.

```
1480 REM SORT FILE
1490 GOSUB 2570
1500 IF L%<2 THEN 1620
1510 CLS:PRINT"Sorting by ";A$(F%,0)
1520 N%=L%
1530 N%=(N%+2)\3
1540 FOR D%=N%+1 TO N%*2
```

```
1550 FOR E%=D% TO L% STEP N%
1560 FOR R%=E% TO D% STEP -N%
1570 IF A$(F%,R%)<A$(F%,R%-N%) THEN FOR
C%=0 TO fields%:K$=A$(C%,R%):A$(C%,R%)=A
$(C%,R%-N%):A$(C%,R%-N%)=K$:NEXT ELSE 15
90
1580 NEXT
1590 NEXT E%
1600 NEXT
1610 IF N%>1 THEN 1530
1620 RETURN
```

Επειδή το επόμενο κεφάλαιο είναι εξ ολοκλήρου αφιερωμένο στο θέμα ταξινόμηση και αναζήτηση, εδώ δεν θα προσπαθήσουμε καθόλου να εξηγήσουμε τις λεπτομέρειες της υπορουτίνας παρά μόνο θα επισημάνουμε ότι χρησιμοποιείται ο αλγόριθμος φθίνοντος βήματος. Αν ληφθεί υπ' όψη ότι είναι σε BASIC, η διαδικασία είναι λογική αλλά, για μεγάλα αρχεία, θα δοθεί στο επόμενο κεφάλαιο μία εναλλακτική υπερ - ταχεία ταξινόμηση σε γλώσσα μηχανής. Η επικεφαλίδα, μέσω της οποίας διεξάγεται η αναζήτηση, λαμβάνεται με την κλήση της GET FIELD HEADING. Ολόκληρο το αρχείο - πίνακας A$(F%, R%) ταξινομείται, με πρώτη την *χαμηλότερη* εγγραφή (κατά αύξουσα σειρά).

Υπορουτίνα ADD RECORDS

Καλείται με τη GOSUB 1060.

```
1050 REM ADD RECORDS
1060 CLS: IF mainsize%>=filesize% OR FRE
(0) < fields%*100 THEN PRINT"File full":
GOSUB 2520:GOTO 1240
1070 mainsize%=mainsize%+1
1080 PRINT" type EXIT to finish entry of
 records"
1090 PRINT:PRINT"Record No :"mainsize%
1100 LOCATE 21,3
1110 PRINT"Bytes Free :"FRE(0)
1120 GOSUB 2330
1130 PRINT:F%=-1
1140 F%=F%+1
1150 PRINT A$(F%,0)
1160 GOSUB 2260:A$(F%,mainsize%)=K$
1170 X%=POS(#0):Y%=VPOS(#0)
1180 LOCATE 34,3
1190 PRINT USING "*****";FRE(0)
1200 LOCATE X%,Y%
```

```
1210 IF A$(F%,mainsize%)="EXIT" THEN mai
nsize%=mainsize%-1:GOTO 1240
1220 IF F%<fields% THEN 1140
1230 IF mainsize%<filesize% THEN 1060
1240 RETURN
```

Ένα αρχείο μεγαλώνει καθώς προστίθενται καινούριες εγγραφές. Αν δεν ενσωματωθούν σχετικοί έλεγχοι, η μνήμη μπορεί ξαφνικά να ξεχειλίσει (overflow) χωρίς προειδοποίηση. Υπάρχουν δύο μηχανισμοί που μπορούν να οδηγήσουν στο σημείο αυτό:

(α) ο αριθμός των εγγραφών μπορεί να φτάσει το όριο που έχει τεθεί από μία υπάρχουσα εντολή DIMension.

(β) Ο συνολικός αριθμός των χρησιμοποιούμενων bytes μπορεί να φτάσει στο όριο που τίθεται από τη διαθέσιμη δυναμικότητα της RAM.

Αν η ποσότητα κειμένου ανά εγγραφή είναι μέτρια, τότε πρώτος περιοριστικός παράγοντας θα είναι ο μέγιστος επιτρεπόμενος αριθμός εγγραφών. Αφ' ετέρου, ένας μέτριος αριθμός εγγραφών, υπερφορτωμένων με κείμενο, μπορεί να προκαλέσει ένα ξεχείλισμα νωρίτερα του αναμενομένου. Αυτή η υπορουτίνα ξεκινάει ελέγχοντας τον τρέχοντα αριθμό εγγραφών που είναι ήδη στο αρχείο και επίσης την ποσότητα της διαθέσιμης RAM αφήνοντας κάποιο ικανοποιητικό συντελεστή ασφαλείας. Η πρώτη γραμμή στην υπορουτίνα πραγματοποιεί και τους δύο ελέγχους. Το ποσό του ορίου ασφαλείας ελεύθερης RAM έχει καθοριστεί αυθαίρετα, μέσω κάποιου σταθερού συντελεστή ασφαλείας, ίσο με 100 X αριθμός των πεδίων. Αν η συνολική ελεύθερη RAM είναι λιγότερη απ' αυτό το σταθερό ποσό, εμφανίζεται το μήνυμα «Αρχείο πλήρες» (file full) και μετά την κλήση της PRESS ANY KEY, η υπόλοιπη υπορουτίνα παραλείπεται μέσω της GOTO 1240.

Οι εγγραφές καταχωρούνται στην A$(F%, R%), μέσω κλήσης της GET LINE INPUT. Τα δεδομένα εισάγονται, ένα πεδίο κάθε φορά, από ένα βρόχο με αυξητικό βήμα στο F%. Αν ανακαλυφθεί η λέξη EXIT, η υπορουτίνα τερματίζεται με την GO TO 1240. Όσο καταχωρούνται οι εγγραφές, εμφανίζονται στο πάνω μέρος της οθόνης ο τρέχων αριθμός εγγραφής και τα υπάρχοντα ελεύθερα bytes.

Υπορουτίνα CREATE FILE

Καλείται με τη GOSUB 390.

```
380 REM CREATE FILE
390 PRINT"Enter file size (number of rec
ords)
400 INPUT filesize%
410 IF filesize%<1 THEN 400
420 PRINT"Enter number of fields require
d (2-15)"
```

```
430 INPUT fields%
440 IF fields%<2 OR fields%>15 THEN 420
450 fields%=fields%-1
460 flag%=1
470 DIM A$(fields%,filesize%)
480 CLS
490 FOR F%=0 TO fields%
500 PRINT"Enter field heading ";F%+1
510 GOSUB 2260:A$(F%,0)=K$
520 NEXT
530 GOSUB 1060
540 RETURN
```

Αυτή η υπορουτίνα αναλαμβάνει την απαραίτητη προκαταρκτική εργασία για τη δημιουργία ενός καινούριου αρχείου. Οι πρώτες δύο μεταβλητές «filesize%» και «fields%» λαμβάνονται άμεσα με το ερώτημα για την εκτίμηση του μέγιστου αριθμού εγγραφών που θα έχει τελικά το αρχείο, και αντίστοιχα για τον αριθμό των πεδίων. Αυτή η πληροφορία τροφοδοτεί τη γραμμή DIMension και ένας δείκτης (flag) παίρνει την τιμή 1, καθορίζοντας την κατάσταση του συγκεκριμένου αρχείου. Οι επικεφαλίδες κάθε πεδίου λαμβάνονται τότε μέσω κλήσης της GET LINE INPUT και καταχωρούνται στην θέση Record0 (εγγραφή) του πίνακα. Οι πρώτες εγγραφές στο νεοφτιαγμένο αρχείο καταχωρούνται τότε με κλήση της ADD RECORDS.

Υπορουτίνα MODIFY/DELETE RECORD

Καλείται με τη GOSUB 1270.

```
1260 REM MODIFY/DELETE RECORD
1270 F%=0:GOSUB 2670:K$=""
1280 WHILE K$<>"EXIT" AND K$<>"KILL"
1290 CLS:PRINT"Mod/copy line with CURSOR
/COPY keys"
1300 PRINT"Type KILL to delete record"
1310 PRINT"Type EXIT to regain option pa
ge"
1320 GOSUB 2330
1330 FOR F%=0 TO fields%
1340 PRINT A$(F%,0) TAB(21) A$(F%,R%)
1350 NEXT
1360 GOSUB 2330:PRINT:GOSUB 2330
1370 LOCATE 1,VPOS(#0)-2
1380 LINE INPUT K$
1390 K$=UPPER$(K$)
1400 IF LEN(K$)>38 THEN K$=LEFT$(K$,38)
1410 FOR F%=0 TO fields%
```

```
1420 IF A$(F%,0)=LEFT$(K$,LEN(A$(F%,0)))
 THEN A$(F%,R%)=RIGHT$(K$,LEN(K$)-20)
1430 NEXT
1440 WEND
1450 IF K$="KILL" THEN WHILE R%<=mainsiz
e%:FOR F%=0 TO fields%:A$(F%,R%)=A$(F%,R
%+1):NEXT:R5=R5+1:WEND:mainsize%=mainsiz
e%-1:PRINT FRE("")
1460 RETURN
```

Για να τροποποιήσουμε μία εγγραφή, πρέπει πρώτα να βρεθεί μέσα στο αρχείο· έτσι αρχικά καλείται η FIND RECORD. Το κύριο μέρος της υπορουτίνας είναι μέσα σ' ένα βρόχο WHILE/WEND που ελέγχει αν ο χειριστής εισάγει «KILL» ή «EXIT». Οι γραμμές 1330 μέχρι 1350 εμφανίζουν την υπό τροποποίηση εγγραφή. Οι δύο επόμενες κλήσεις της DRAW LINE κρατάνε χώρο στην οθόνη για την εισαγωγή των τροποποιούμενων χαρακτήρων που, με τη βοήθεια του VPOS στη γραμμή 1370, παρουσιάζονται στο διάστημα ανάμεσα στις δύο γραμμές. Αν ο χειριστής εισάγει το KILL, η γραμμή 1450 διαγράφει την εγγραφή και αριθμεί πάλι αυτές που ακολουθούν για να κλείσει το χάσμα. Για περισσότερες λεπτομέρειες για την ιδιαίτερη χρήση αυτής της υπορουτίνας, βλέπε στο Κεφάλαιο 4.

Υπορουτίνα TITLE

Καλείται με τη GOSUB 3260.

```
3250 REM TITLE
3260 CLS
3270 LOCATE 1,2
3280 INK 2,6
3290 PAPER 2
3300 PRINT" MULTIFILING SYSTEM "
3310 PAPER 0
3320 RETURN
```

Αυτή, όπως και η υπορουτίνα για τη σχεδίαση γραμμής, είναι ασήμαντη αλλά μας γλυτώνει απ' το να εισάγουμε μία γραμμή τίτλου στο πάνω μέρος της οθόνης. Είναι απλή υπόθεση να αλλάξετε τις τιμές για το πλαίσιο, το μελάνι (ink) και τη γραφίδα (pen) αν αισθάνεστε ότι δεν έχει γούστο η επιλογή χρωμάτων του συγγραφέα.

Υπορουτίνα SEARCH FILE MENU

Καλείται με τη GOSUB 1650.

```
1640 REM SEARCH FILE MENU
1650 SEL%=0:GOSUB 3260
```

```
1660 LOCATE 1,5
1670 WHILE SEL%<1 OR SEL%>12
1680 PRINT"SHRINK file (Non destructive)
"
1690 PRINT"SEARCH any field for:":PRINT
1700 PRINT"(1)  <> a"
1710 PRINT"(2)   = a"
1720 PRINR"(3)  >= a"
1730 PRINT"(4)  <  a"
1740 PRINT"(5)  Character group"
1750 PRINT:PRINT"Further subfile options
:":PRINT
1760 PRINT"(6)  Save sub-file"
1770 PRINT"(7)  Display sub-file"
1780 PRINT"(8)  Sort sub-file by any fie
ld"
1790 PRINT"(9)  Delete sub-file from mai
n file"
1800 PRINT"(10) Totalise column"
1810 PRINT"(11) Print sub-file"
1820 PRINT"(12) Return to main menu"
1830 LOCATE 1,24
1840 INPUT"Select option ";K$
1850 SEL%=VAL(K$)
1860 CLS
1870 WEND
1880 CLS:L%=subsize%
1890 ON SEL% GOSUB 2770,2770,2770,2770,2
770,700,820,1490,3170,2150,1940,1910
1900 IF SEL%<12 THEN 1650
1910 RETURN
```

Αυτός είναι ένας βοηθητικός κατάλογος επιλογών (menu), που μπορεί να κληθεί όπως μία από τις επιλογές που παρέχονται από ένα βασικό κατάλογο επιλογών. Η πρώτη κλήση γίνεται στην TITLE για την επικεφαλίδα της σελίδας καταλόγου επιλογών. Πολλοί από τους αριθμούς των GOSUB που καλούνται στη γραμμή 1890 ανταποκρίνονται σε υπορουτίνες που έχουν ήδη περιγραφεί. Οι επιλογές 1 μέχρι 4 επιτρέπουν σ' ένα βασικό αρχείο να περιοριστεί σ' ένα υποαρχείο που να περιέχει μόνο εγγραφές που ικανοποιούν ορισμένα κριτήρια καθοριζόμενα από το (α). Παρατηρήστε ότι η εκλογή οποιασδήποτε απ' αυτές τις τέσσερις προκαλεί την κλήση της GOSUB με τον ίδιο αριθμό, τον 2770. Αυτή είναι η υπορουτίνα SEARCH DIRECTOR που θα περιγραφτεί στη συνέχεια. Η επιλογή 5 προϋποθέτει ότι το κριτήριο ένταξης στο υποαρχείο είναι η παρουσία ενός συγκεκριμένου χαρακτήρα ή ομάδας χαρακτήρων.

Υπορουτίνα SEARCH DIRECTOR

Καλείται με τη GOSUB 2770.

```
2760 REM SEARCH DIRECTOR
2770 GOSUB 2570
2780 PRINT"SEARCH to operate on ";A$(F%,
0)
2790 IF SEL%<5 THEN PRINT"Give search da
tum/entity"
2800 IF SEL%=5 THEN PRINT"Give character
 group"
2810 GOSUB 2260:comp$=k$
2820 count%=0
2830 FOR R%=1 TO subsize%
2840 ON SEL% GOSUB 2900,2940,2980,3020,3
060
2850 NEXT
2860 IF count%=0 THEN CLS:PRINT"SEARCH i
s NEGATIVE":GOSUB 2520 ELSE subsize%=cou
nt%
2870 RETURN
2880 '
2890 REM SEARCH <>
2900 IF A$(F%,R%)<>comp$ THEN GOSUB 3100
2910 RETURN
2920 '
2930 REM SEARCH =
2940 IF A$(F%,R%)=comp$ THEN GOSUB 3100
2950 RETURN
2960 '
2970 REM SEARCH >=
2980 IF A$(F%,R%)>=comp$ THEN GOSUB 3100
2990 RETURN
3000 '
3010 REM SEARCH <
3020 IF A$(F%,R%)<comp$ THEN GOSUB 3100
3030 RETURN
3040 '
3050 REM SEARCH FOR CHARACTER GROUP
3060 IF INSTR(A$(F%,R%),comp$) <>0 THEN
GOSUB 3100
3070 RETURN
3080 '
3090 REM MOVE RECORD
```

```
3100 count%=count%+1
3110 FOR C%=0 TO fields%
3120 K$=A$(C%,R%):A$(C%,R%)=A$(C%,count%
):A$(c%,count%)=K$
3130 NEXT
3140 RETURN
```

Για τη δημιουργία ενός υποαρχείου, ο χειριστής πρέπει να εισάγει την *επικεφαλίδα πεδίου* που τον ενδιαφέρει, έτσι πρώτα καλείται η FIND FIELD HEADING. Αν SEL% = 5. (SEL% είναι παράμετρος που μεταβιβάζεται από τον κατάλογο επιλογών υποαρχείου) εμφανίζονται υπενθυμίσεις για την τιμή του (α), ή μιας ομάδας χαρακτήρων. Η απάντηση καταχωρείται στην comp$ καλώντας την GET LINE INPUT. Η μεταβλητή comp$ λέγεται έτσι γιατί περιέχει την παράμετρο που θα χρησιμοποιηθεί για σύγκριση (comparison) με κάθε εγγραφή του βασικού αρχείου. Μία χωριστή υπορουτίνα για κάθε μία από τις συγκρίσεις καλείται μέσα από το βρόχο FOR/NEXT στις γραμμές 2830 μέχρι 2850. Αν και ο βρόχος εκτείνεται από R% = 1 μέχρι subsize%, καλύπτει το σύνολο του βασικού αρχείου με την πρώτη κλήση, γιατί η subsize% θα έχει πάρει προηγουμένως την τιμή mainsize%. Όλες οι εγγραφές που ταιριάζουν με τα εισαγόμενα δεδομένα ως προς την comp$ μεταφέρονται στην *αρχή του βασικού αρχείου* με τη βοήθεια της υπορουτίνας MOVE RECORD στη γραμμή 3100. Επειδή το υποαρχείο είναι η συλλογή των εγγραφών που βρίσκονται τώρα στην αρχή (μπροστινό μέρος) του βασικού αρχείου, δεν καταλαμβάνεται καθόλου πρόσθετος χώρος RAM. Η μεταβλητή, subsize%, κρατάει τον αριθμό εγγραφών του υποαρχείου.

Υπορουτίνα DELETE SUBFILE

```
3160 REM DELETE SUB FILE
3170 IF subsize%=mainsize% THEN 3220
3180 FOR R%=subsize%+1 TO mainsize%
3190 FOR F%=0 TO fields%

3200 A$(F%,R%-subsize%)=A$(F%,R%)
3210 NEXT:NEXT
3220 mainsize%=mainsize%-subsize%:SEL%=1
2
3230 RETURN
```

Αυτή η υπορουτίνα καλείται μόνο αν το υποαρχείο πρόκειται να διαγραφεί από το βασικό αρχείο. Η γραμμή 3180 περιέχει το νήμα που οδηγεί στον τρόπο λειτουργίας της γιατί η παράμετρος έναρξης του βρόχου FOR / NEXT είναι subsize%+1. Μ' άλλα λόγια, το βασικό αρχείο τώρα αρχίζει από τον πρώτο αριθμό εγγραφής που είναι μεγαλύτερος από τον τελευταίο (αριθμό εγγραφής) του υποαρχείου. (Θα θυμάστε ότι οι εγγραφές, οι συγκεντρωμένες σ' ένα υποαρχείο, είχαν όλες επανατοποθετηθει στην κορυφή του βασικού αρχείου).

Επαναφορά στην αρχική κατάσταση και βασικός κατάλογος επιλογών (main menu)

```
10 REM INITIALISATION AND MAIN MENU
20 OPENOUT "buffer"
30 MEMORY &9B7E:REM SET TO HIMEM-1
40 CLOSEOUT
50 MODE 1
60 BORDER 0
70 mainsize%=0:flag%=0
80 SEL%=0
90 WHILE SEL%<1 OR SEL%>12
100 GOSUB 3260
110 LOCATE 5,5
120 PRINT"MAIN FILE MENU"
130 LOCATE 1,7
140 PRINT"(1)  Create new file"
150 PRINT"(2)  Load file"
160 PRINT"(3)  Save file"
170 PRINT"(4)  Display file"
180 PRINT"(5)  Add records"
190 PRINT"(6)  Modify any record"
200 PRINT"(7)  Sort by any field"
210 PRINT"(8)  Create sub-file (Search a
ny field)"
220 PRINT"(9)  Print File"
230 PRINT"(10) Display single record"
240 PRINT"(11) Totalise any column"
250 PRINT"(12) End program"
260 LOCATE 1,23
270 INPUT"Select option ";K$
280 SEL%=VAL(K$)
290 WEND
300 CLS:L%=mainsize%:subsize%=mainsize%
310 IF SEL%>2 AND mainsize%=0 AND SEL%<>
12 THEN PRINT"No file present":GOSUB 252
0:GOTO 70
320 IF SEL%<3 AND flag%=1 THEN 2440
330 ON SEL% GOSUB 390,570,700,820,1060,1
270,1490,1650,1940,2040,2150,350
340 SPEED WRITE 0
360 END
```

Αυτό είναι ένα πρόγραμμα ελέγχου, όχι υπορουτίνα. Έχει τρία τμήματα:

(1) Οι γραμμές 10 μέχρι 80 επαναφέρουν στην αρχική κατάσταση, καθορίζουν μία *μόνιμη* περιοχή στη μνήμη για τον buffer της κασέτας, το mode και το χρώμα του φόντου.

(2) Οι γραμμές 90 μέχρι 290 καταλαμβάνονται από ένα βρόχο WHILE / WEND που περιέχει το βασικό κατάλογο επιλογών και την υπενθύμιση στο χειριστή να δώσει έναν αριθμό επιλογής στη SEL%. Ο τίτλος στο πάνω μέρος του εμφανιζόμενου κατάλογου επιλογών έρχεται μέσω κλήσης της υπορουτίνας TITLE.

(3) Οι γραμμές 300 μέχρι 360 ξαναδίνουν τιμές στην L% (το τρέχον μήκος του αρχείου) και στη subsize% (το μέγεθος του υποαρχείου). Η ισχύς της εκλογής του χειριστή ελέγχεται κατόπιν καλώντας την κατάλληλη υπορουτίνα. Τελικά επαναφέρεται σε ισχύ η λανθάνουσα ταχύτητα baud σε περίπτωση που μία κλήση της υπορουτίνας SAVE FILE την έχει αφήσει 2000 baud.

Πίνακας αριθμών υπορουτινών.

Οι υπορουτίνες που δόθηκαν σ' αυτό το κεφάλαιο σαν δομικά στοιχεία γενικής χρήσης είναι επίσης συστατικά μέρη ενός ολοκληρωμένου προγράμματος διαχείρισης αρχείων. Είναι γεγονός ότι οι αριθμοί γραμμών των υπορουτινών θα διαπιστωθεί ότι ανταποκρίνονται ακριβώς στην λίστα του Προγράμματος 4.1. στο επόμενο κεφάλαιο. Επειδή η ανάλυση κάθε υπορουτίνας έχει περιγραφεί εδώ σε λογικό βαθμό, το Κεφάλαιο 4 θα επικεντρωθεί στις οδηγίες για τη *χρήση* του προγράμματος. Για λόγους παραπομπής, οι αριθμοί γραμμών υπορουτινών εκτίθενται παρακάτω. Αξίζει να αναφερθεί σ' αυτό το σημείο ότι τα GOTO και τα GOSUB σε γραμμές με δηλώσεις REM είναι κακή τακτική. Ο λόγος είναι ότι οι *συμπυκνωτές (compactors)* του προγράμματος μετακινούν όλες μαζί τις γραμμές REM και είναι αδύνατο να βρουν που να εκχωρήσουν τις GOTO και GOSUB. Αυτό κάνει επίσης ευκολότερη την αφαίρεση των REM από το χαρτί για κάποιο δοκιμαστικό αντίγραφο, όπου απαιτείται περισσότερος χώρος RAM.

Πίνακας 3.1. Πίνακας των αριθμών GOSUB (αγνοούνται οι πρώτες γραμμές REM).

Αριθμός γραμμής	Τίτλος GOSUB
233ϕ	DRAW LINE
226ϕ	GET LINE INPUT
252ϕ	PRESS ANY KEY
237ϕ	GET FILENAME
7ϕϕ	SAVE FILE
57ϕ	LOAD FILE

82ϕ	DISPLAY FILE
267ϕ	FIND RECORD
2ϕ4ϕ	DISPLAY RECORD
257ϕ	FIND FIELD HEADING
194ϕ	PRINT FILE
215ϕ	TOTALISE COLUMN
149ϕ	SORT FILE
1ϕ6ϕ	ADD RECORS
39ϕ	CREATE FILE
127ϕ	MODIFY/DELETE RECORD
326ϕ	TITLE
165ϕ	SEARCH FILE MENU
277ϕ	SEARCH FILE DIRECTOR
317ϕ	DELETE SUBFILE

Περίληψη

1. Είναι κακή τακτική το άλμα έξω από μία υπορουτίνα πριν το κανονικό RETURN.
2. Ένα αρχείο «βασισμένο στο πρόγραμμα» έχει εγγραφές μέσα σε εντολές τύπου DATA.
3. Η A$(F%, R%) είναι στοιχείο ενός πίνακα βασικού αρχείου, ή πεδίο στην περίπτωσή μας.
4. Ο δείκτης πληροφορίας πεδίου, F%, γράφεται κατά τέτοιο τρόπο ώστε να συμφωνεί πρώτα με την τάξη των bytes περιγραφής της αλφαριθμητικής παράστασης που αποθηκεύονται στη μνήμη από τον interpreter.
5. Στο πρόγραμμα 3.2 η A$(0,R%) είναι το πεδίο-κλειδί της εγγραφής R%.
6. Τα αρχεία δεδομένων φυλάγονται στην ταινία με διάταξη τελείως διαφορετική από τα αρχεία προγραμμάτων.
7. Η κατεύθυνση προς την κασετοταινία ή τη δισκέτα ορίζεται σαν κανάλι 9.
8. Η εντολή SPEEDWRITE 1 χρησιμοποιείται αν τα δεδομένα πρόκειται να καταγραφούν με (ταχύτητα) 2000 baud.
9. Ένα αρχείο πρέπει κανονικά να κλείνεται με την κατάλληλη λέξη (keyword), CLOSEOUT ή CLOSEIN.
10. Η παράλειψη κλεισίματος ενός αρχείου για έξοδο δεδομένων θα μπορούσε να έχει σαν αποτέλεσμα τα παραμένοντα δεδομένα στον buffer να μην πάνε στο αρχείο.
11. Η K$ είναι μία προσωρινή μεταβλητή.
12. Η υπορουτίνα GET LINE INPUT χρησιμοποιείται για την είσοδο αλφαριθμητικών μεταβλητών. Απορρίπτει την κενή αλφαριθμητική παράσταση (null string) και αποκόβει όλα τα εισαγόμενα στοιχεία σε 18 χαρακτήρες.
13. Οι αριθμοί εγγραφών που ακολουθούν μία διαγραμμένη εγγραφή αλ-

λάζουν όλοι κατά 1.

14. Όταν οι εγγραφές συγκεντρώνονται για να αποτελέσουν ένα υποαρχείο, ξανατοποθετούνται στην αρχή (μπροστινό τμήμα) του βασικού αρχείου.

15. Οι αριθμοί των GOSUB στα δομικά στοιχεία είναι κανονισμένοι να προσπερνάνε τα αρχικά REM.

Ασκήσεις

3.1. Αν προστεθούν άλλες πέντε εγγραφές στο πρόγραμμα ποιά άλλη γραμμή πρέπει να τροποποιηθεί;

3.2. Ποιός είναι ο σκοπός του flag% στο Πρόγραμμα 3.2;

3.3. Στο Πρόγραμμα 3.2., ποιός είναι ο σκοπός της γραμμής 310;

3.4. Πώς αιτιολογείται η γραμμή 1500 στην υπορουτίνα SORT FILE;

Κεφάλαιο Τέσσερα

Ένα Ολοκληρωμένο Σύστημα Διαχείρισης Σειριακού Αρχείου Βασισμένο στη RAM

Η λίστα του προγράμματος

Αυτό το κεφάλαιο είναι εξ ολοκλήρου αφιερωμένο στο χειρισμό και την περιγραφή ενός ολοκληρωμένου προγράμματος διαχείρισης αρχείου βασισμένου στη RAM.

```
10 REM CASSETTE MULTIFILING SYSTEM
20 OPENOUT "buffer"
30 MEMORY &9B7E:REM SET TO HIMEM-1
40 CLOSEOUT
50 MODE 1
60 BORDER 0
70 mainsize%=0:flag%=0
80 SEL%=0
90 WHILE SEL%<1 OR SEL%>12
100 GOSUB 3260
110 LOCATE 5,5
120 PRINT"MAIN FILE MENU"
130 LOCATE 1,7
140 PRINT"(1)  Create new file"
150 PRINT"(2)  Load file"
160 PRINT"(3)  Save file"
170 PRINT"(4)  Display file"
180 PRINT"(5)  Add records"
190 PRINT"(6)  Modify any record"
200 PRINT"(7)  Sort by any field"
```

```
210 PRINT"(8)  Create sub-file (Search any field)"
220 PRINT"(9)  Print File"
230 PRINT"(10) Display single record"
240 PRINT"(11) Totalise any column"
250 PRINT"(12) End program"
260 LOCATE 1,23
270 INPUT"Select option ";k$
280 SEL%=VAL(K$)
290 WEND
300 CLS:L%=mainsize%:subsize%=mainsize%
310 IF SEL%>2 AND mainsize%=0 AND SEL%<>
12 THEN PRINT"No file present":GOSUB 252
0:GOTO70
320 IF SEL%<3 AND flag%=1 THEN 2440
330 ON SEL% GOSUB 390,570,700,820,1060,1
270,1490,1650,1940,2040,2150,350
340 GOTO 80
350 SPEED WRITE 0
360 END
370 '
380 REM CREATE FILE
390 PRINT"Enter file size (number of rec
ords)"
400 INPUT filesize%
410 IF filesize%<1 THEN 400
420 PRINT"Enter number of fields require
d (2-15)"
430 INPUT fields%
440 IF fields%<2 OR fields%>15 THEN 420
450 fields%=fields%-1
460 flag%=1
470 DIM A$(fields%,filesize%)
480 CLS
490 FOR F%=0 TO fields%
500 PRINT"Enter field heading ";F%+1
510 GOSUB 2260:A$(F%,0)=K$
520 NEXT
530 GOSUB 1060
540 RETURN
550 '
560 REM LOAD FILE
570 GOSUB 2370
580 OPENIN filename$
590 INPUT#9,filesize%,fields%,mainsize%
600 flag%=1
```

```
610 DIM A$(fields%,filesize%)
620 FOR R%=0 TO mainsize%
630 FOR F%=0 TO fields%
640 INPUT#9,A$(F%,R%)
650 NEXT:NEXT
660 CLOSEIN
670 RETURN
680 '
690 REM SAVE FILE
700 GOSUB 2370
710 SPEED WRITE 1
720 OPENOUT filename$
730 PRINT#9,filesize%,fields%,L%
740 FOR R%=0 TO L%
750 FOR F%=0 TO fields%
760 PRINT#9,A$(F%,R%)
770 NEXT:NEXT
780 CLOSEOUT
790 RETURN
800 '
810 REM DISPLAY FILE
820 F%=1:top%=1
830 WHILE INKEY(47)<>0
840 CLS:PRINT"Press space bar to regain
menu"
850 GOSUB 2330
860 PRINT A$(0,0) TAB(20) A$(F%,0)
870 GOSUB 2330:bottom%=top%+19
880 IF bottom%>L% THEN bottom%=L%
890 FOR R%=top% TO bottom%
900 PRINT A$(0,R%) TAB(20) A$(F%,R%)
910 NEXT
920 CALL &BB03:'KM RESET
930 CALL &BB18:'KM WAIT KEY
940 IF INKEY(8)=0 THEN F%=F%-1
950 IF INKEY(1)=0 THEN F%=F%+1
960 IF INKEY(0)=0 THEN top%=top%-20
970 IF INKEY(2)=0 THEN top%=top%+20
980 IF F%<1 THEN F%=fields%
990 IF F%>fields% THEN F%=1
1000 IF top%<1 THEN top%=(INT(L%/20)*20)
+1
1010 IF top%>L% THEN top%=1
1020 WEND
1030 RETURN
```

```
1040 '
1050 REM ADD RECORDS
1060 CLS:IF mainsize%>=filesize% OR FRE(
0)<fields%*100 THEN PRINT"File full":GOS
UB 2520:GOTO 1240
1070 mainsize%=mainsize%+1
1080 PRINT"Type EXIT to finish entry of
records"
1090 PRINT:PRINT"Record No :"mainsize%
1100 LOCATE 21,3
1110 PRINT"Bytes free :"FRE(0)
1120 GOSUB 2330
1130 PRINT:F%=-1
1140 F%=F%+1
1150 PRINT A$(F%,0)
1160 GOSUB 2260:A$(F%,mainsize%)=K$
1170 X%=POS(#0):Y%=VPOS(#0)
1180 LOCATE 34,3
1190 PRINT USING "#####";FRE(0)
1200 LOCATE X%,Y%
1210 IF A$(F%,Zmainsize%)="EXIT" THEN ma
insize%=mainsize%-1:GOTO 1240
1220 IF F%<fields% THEN 1140
1230 IF mainsize%<filesize% THEN 1060
1240 RETURN
1250 '
1260 REM MODIFY/DELETE RECORD
1270 F%=0:GOSUB 2670:K$=""
1280 WHILE K$<>"EXIT" AND K$<>"KILL"
1290 CLS:PRINT"Mod/copy line with CURSOR
/COPY keys"
1300 PRINT"Type KILL to delete record"
1310 PRINT"Type EXIT to regain option pa
ge"
1320 GOSUB 2330
1330 FOR F%=0 TO fields%
1340 PRINT A$(F%,0) TAB(21) A$(F%,R%)
1350 NEXT
1360 GOSUB 2330:PRINT:GOSUB 2330
1370 LOCATE 1,VPOS(#0)-2
1380 LINE INPUT K$
1390 K$=UPPER$(K$)
1400 IF LEN(K$)>38 THEN K$=LEFT$(K$,38)
1410 FOR F%=0 TO fields%
```

```
1420 IF A$(F%,0)=LEFT$(K$,LEN(A$(F%,0)))
 THEN A$(F%,R%)=RIGHT$(K$,LEN(K$)-20)
1430 NEXT
1440 WEND
1450 IF K$="KILL" THEN WHILE R%<=mainsiz
e%:FOR F%=0 TO fields%:A$(F%,R%)=A$(F%,R
%+1):NEXT:R%=R%+1:WEND:mainsize%=mainsiz
e%-1:PRINT FRE("")
1460 RETURN
1470 '
1480 REM SORT FILE
1490 GOSUB 2570
1500 IF L%<2 THEN 1620
1510 CLS:PRINT"Sorting by ";A$(F%,0)
1520 N%=L%
1530 N%=(N%+2)/3
1540 FOR D%=N%+1 TO N%*2
1550 FOR E%=D% TO L% STEP N%
1560 FOR R%=E% TO D% STEP -N%
1570 IF A$(F%,R%)<A$(F%,R%-N%) THEN FOR
C%=0 TO fields%:K$=A$(C%,R%):A$(C%,R%)=A
$(C%,R%-N%):A$(C%,R%-N%)=K$:NEXT ELSE 15
90
1580 NEXT
1590 NEXT E%
1600 NEXT
1610 IF N%>1 THEN 1530
1620 RETURN
1630 '
1640 REM SEARCH FILE MENU
1650 SEL%=0:GOSUB 3260
1660 LOCATE 1,5
1670 WHILE SEL%<1 OR SEL%>12
1680 PRINT"SHRINK file (Non destructive)
"
1690 PRINT"SEARCH any field for :":PRINT
1700 PRINT"(1) <>a"
1710 PRINT"(2)  =a"
1720 PRINT"(3) >=a"
1730 PRINT"(4) < a"
1740 PRINT"(5) character group"
1750 PRINT:PRINT"Further subfile options
 :":PRINT
```

```
1760 PRINT"(6) Save sub-file"
1770 PRINT"(7) Display sub-file"
1780 PRINT"(8) Sort sub-file by any fiel
d"
1790 PRINT"(9) Delete sub-file from main
 file"
1800 PRINT"(10)Totalise column"
1810 PRINT"(11)Print sub-file"
1820 PRINT"(12)Return to main menu"
1830 LOCATE 1,24
1840 INPUT"Select option ";K$
1850 SEL%=VAL(K$)
1860 CLS
1870 WEND
1880 CLS:L%=SUBSIZE%
1890 ON SEL% GOSUB 2770,2770,2770,2770,2
770,700,820,1490,3170,2150,1940,1910
1900 IF SEL%<12 THEN 1650
1910 RETURN
1920 '
1930 REM PRINT FILE
1940 FOR R%=1 TO L%
1950 PRINT#8:PRINT#8,"Record No :"R%
1960 PRINT#8,STRING$(40,"-")
1970 FOR F%=0 TO fields%
1980 PRINT#8,A$(F%,0) TAB(21) A$(F%,R%)
1990 NEXT
2000 NEXT
2010 RETURN
2020 '
2030 REM DISPLAY RECORD
2040 F%=0:GOSUB 2670:CLS
2050 PRINT:PRINT"Current record number"R
%:PRINT
2060 GOSUB 2330
2070 FOR F%=0 TO fields%
2080 PRINT A$(F%,0) TAB(21) A$(F%,R%)
2090 NEXT
2100 GOSUB 2330
2110 GOSUB 2520
2120 RETURN
2130 '
2140 REM TOTALISE COLUMN
```

```
2150 total=0
2160 GOSUB 2570
2170 FOR R%=1 TO L%
2180 total=total+VAL(A$(F%,R%))
2190 NEXT
2200 PRINT"Column total ="total
2210 PRINT"Column average="total/L%
2220 GOSUB 2520
2230 RETURN
2240 '
2250 REM GET LINE INPUT
2260 LINE INPUT K$
2270 IF K$="" THEN 2260
2280 K$=UPPER$(K$)
2290 IF LEN(K$)>18 THEN K$=LEFT$(K$,18)
2300 RETURN
2310 '
2320 REM DRAW LINE
2330 PRINT STRING$(40,CHR$(154));
2340 RETURN
2350 '
2360 REM GET FILENAME
2370 CLS:PRINT"Enter filename"
2380 GOSUB 2260
2390 IF LEN(K$)>16 THEN PRINT"Too long:E
nter again":GOTO 2380
2400 filename$=K$
2410 RETURN
2420 '
2430 REM BELT & BRACES
2440 PRINT"WARNING!":SOUND 1,16,10,15
2450 PRINT"This selection destroys exist
ing file"
2460 PRINT"Press ANY KEY to CANCEL selec
tion"
2470 PRINT"OR press CTRL M keys and RESE
LECT"
2480 CALL &BB18:'KM WAIT KEY
2490 IF INKEY(38)=128 THEN ERASE A$:GOTO
 70 ELSE 80
2500 '
2510 REM PRESS ANY KEY
2520 PRINT:PRINT"Press any key to contin
ue"
```

```
2530 K$=INKEY$:IF K$="" THEN 2530
2540 RETURN
2550 '
2560 REM FINF FIELD HEADING
2570 F%=-1
2580 PRINT"Operate on which field? (Give
 heading)
2590 GOSUB 2260
2600 F%=F%+1
2610 IF K$=LEFT$(A$(F%,0),LEN(K$)) THEN
2640
2620 IF F%<fields% THEN 2600
2630 PRINT"No such field":GOTO 2570
2640 RETURN
2650 '
2660 REM FIND RECORD
2670 R%=0
2680 PRINT"Give record entry under ";a$(
f%,0)
2690 GOSUB 2260
2700 R%=R%+1
2710 IF K$=LEFT$(A$(F%,R%),LEN(K$)) THEN
 2740
2720 IF R%<L% THEN 2700
2730 PRINT"No such record on file":GOTO
2670
2740 RETURN
2750 '
2760 REM SEARCH DIRECTOR
2770 GOSUB 2570
2780 PRINT"SEARCH to operate on ";A$(F%,
0)
2790 IF SEL%<5 THEN PRINT"Give search da
tum/entity"
2800 IF SEL%=5 THEN PRINT"Give character
 group"
2810 GOSUB 2260:comp$=K$
2820 count%=0
2830 FOR R%=1 TO subsize%
2840 ON SEL% GOSUB 2900,2940,2980,3020,3
060
2850 NEXT
```

```
2860 IF count%=0 THEN CLS:PRINT"SEARCH i
s NEGATIVE":GOSUB 2520 ELSE subsize%=cou
nt%
2870 RETURN
2880 '
2890 REM SEARCH <>
2900 IF A$(F%,R%)<>comp$ THEN GOSUB 3100

2910 RETURN
2920 '
2930 REM SEARCH =
2940 IF A$(F%,R%)=comp$ THEN GOSUB 3100
2950 RETURN
2960 '
2970 REM SEARCH >=
2980 IF A$(F%,R%)>=comp$ THEN GOSUB 3100

2990 RETURN
3000 '
3010 REM SEARCH <
3020 IF A$(F%,R%)<comp$ THEN GOSUB 3100
3030 RETURN
3040 '
3050 REM SEARCH FOR CHARACTER GROUP
3060 IF INSTR(A$(F%,R%),comp$)<>0 THEN G
OSUB 3100
3070 RETURN
3080 '
3090 REM MOVE RECORD
3100 count%=count%+1
3110 FOR C%=0 TO fields%
3120 K$=A$(C%,R%):A$(C%,R%)=A$(C%,count%
):A$(C%,count%)=K$
3130 NEXT
3140 RETURN
3150 '
3160 REM DELETE SUB FILE
3170 IF subsize%=mainsize% THEN 3220
3180 FOR R%=subsize%+1 TO mainsize%
3190 FOR F%=0 TO fields%
3200 A$(F%,R%-subsize%)=A$(F%,R%)
3210 NEXT:NEXT
3220 mainsize%=mainsize%-subsize%:SEL%=1
2
3230 RETURN
```

```
3240 '
3250 REM title
3260 CLS
3270 LOCATE 1,2
3280 INK 2,6
3290 PAPER 2
3300 PRINT"MULTIFILING SYSTEM"
3310 PAPER 0
3320 RETURN
```

Πρόγραμμα 4.1. Σύστημα πολυαρχείου βασισμένο στη RAM.

Όπως μπορούμε να δούμε από το μήκος της λίστας χρειάζεται κάμποσος χρόνος για την πληκτρολόγηση κι ίσως ακόμη περισσότερος για τη διόρθωση των σφαλμάτων πληκτρολόγησης. Πριν συνεχίσουμε, αξίζει να επισημάνουμε τη διαφορά μεταξύ «λάθους προγραμματισμού» (bug) και λάθους πληκτρολόγησης. Το λάθος κατά τον προγραμματισμό (bug) προκαλεί στο πρόγραμμα «συμπεριφορά» διαφορετική από αυτή που είχε στο νου του ο προγραμματιστής, με αυτή δε την έννοια δεν υπάρχουν λάθη προγραμματισμού στο Πρόγραμμα 4.1. Αν πληκτρολογήσετε τη λίστα ακριβώς όπως την βλέπετε, το πρόγραμμα θα «δουλέψει» με τον τρόπο που το έχουμε προσχεδιάσει. Αφού περάσει κάποιο διάστημα που θα το έχετε χρησιμοποιήσει, είναι ενδεχόμενο να σκεφθείτε ότι ορισμένα επιμέρους στοιχεία θα μπορούσαν να βελτιωθούν ή να αλλάξουν, ώστε να ταιριάζουν με τις δικές σας ειδικές ανάγκες αρχειοθέτησης. Ενδέχεται ακόμη να μη χρειάζεστε όλες τις εναλλακτικές δυνατότητες (options), περίπτωση κατά την οποία μπορείτε να παραλείψετε μία ή δύο απ' αυτές, ώστε να εξοικονομήσετε περισσότερο χώρο για εγγραφές. Ευτυχώς το πρόγραμμα έχει μία καλοφτιαγμένη λογική δομή και έτσι είναι εύκολο να εισαγάγετε τροποποιήσεις ή να παραλείψετε μερικές επιλογές. Παρ' όλ' αυτά, εάν νομίζετε ότι κάποιες υπορουτίνες μπορούν να συντομευθούν ή να απλουστευθούν, προσέξτε να κρατήσετε την αρχική λίστα στην κασσέτα, ώστε να μπορείτε να επανέλθετε σ' αυτήν σε κάθε ενδεχόμενο. Παράξενα πράγματα μπορεί να συμβούν, όταν κάποιος προσπαθεί να «απλοποιήσει» ένα πρόγραμμα.

Εκείνοι που δεν είναι ακόμη συνηθισμένοι σε μαραθώνιους αγώνες πληκτρολόγησης, ίσως ενδιαφέρονται να μάθουν ότι υπάρχουν δύο τρόποι για να πετύχουν τον σκοπό τους:

(α) Να βάλουν όλα τους τα δυνατά ελπίζοντας για το καλύτερο - το οποίο σημαίνει, ν' αρχίσουν από τη γραμμή 1 και να συνεχίσουν μέχρι την τελευταία γραμμή. Αυτή η μέθοδος οφείλει να αποδόσει - τελικά!

(β) Να παρατηρήσουν ότι το «πρόγραμμα» αυτό καθ' εαυτό, τελειώνοντας με το END, καταλαμβάνει περίπου τις πρώτες 40 γραμμές μόνο· το υπόλοιπο της λίστας είναι μία συλλογή από υπορουτίνες. Αυτές οι γραμμές (του κυρίως προγράμματος) θα πρέπει να πληκτρολογηθούν πρώτα και μετά όλες οι υπορουτίνες που είναι αναγκαίες, για να βγει σε πέρας η επιλογή 1, «Crea-

te file» (Δημιουργία αρχείου). Θυμηθείτε ότι η επιλογή 1 ανακαλεί και μερικές άλλες υπορουτίνες, έτσι, φυσικά, θα πρέπει να τις εισάγετε κι αυτές. Θα σας βοηθούσε εάν μελετούσατε τον Πίνακα 4.1 που παρουσιάζεται αργότερα σ' αυτό το κεφάλαιο. Μπορείτε εδώ να ξεκινήσετε δημιουργώντας ένα μικρό αρχείο τριών εγγραφών, με τρία πεδία στην καθεμία, όπως: Όνομα, Ηλικία, Επάγγελμα. Εργαστείτε με τη σειρά μ' αυτές τις γραμμές για κάθε επιλογή για να βεβαιωθείτε ότι το πρόγραμμα τρέχει ικανοποιητικά μέχρι το σημείο που έχετε φθάσει. Πάντως υπάρχει τεράστια διαφορά ανάμεσα σ' ένα πρόγραμμα που φαίνεται ότι δουλεύει ικανοποιητικά και σ' ένα που έχει υποβληθεί σε εξαντλητικά τεστ. Αυτούς τους ελέγχους πάντως θα πρέπει να τους αφήσετε γι' αργότερα, όταν θα έχετε ολοκληρώσει την εξερεύνηση όλου του προγράμματος.

Χρήση του προγράμματος

Λεπτομέρειες για τον προγραμματισμό δίνονται στο τέλος αυτού του κεφαλαίου γι' αυτούς που μπορεί να θέλουν να εισάγουν τροποποιήσεις ή να προσθέσουν συμπληρωματικές δυνατότητες. Παρ' όλα αυτά, θα ήταν φρόνιμο να ξοδέψετε λίγο χρόνο βάζοντας το πρόγραμμα να λειτουργήσει, όπως είναι. Τρέχοντας το πρόγραμμα εμφανίζεται ο κατάλογος επιλογών (menu) με τις 12 εναλλακτικές δυνατότητες. Αρχικά δεν υπάρχει κανένα αρχείο στην RAM έτσι υπάρχουν μόνο δύο δυνατές επιλογές απ' τον κατάλογο - η Επιλογή 1 «Create file» (Δημιουργία αρχείου) ή η Επιλογή 2 «Load file» (Φόρτωση αρχείου). Αν διαλέξετε κάποια άλλη επιλογή, εμφανίζεται το μήνυμα «No file present», δηλαδή δεν υπάρχει κανένα αρχείο στην RAM. Το menu ξαναεμφανίζεται στην οθόνη εάν απαντήσετε στην υπενθύμιση «Press any key to continue», δηλ. «Πάτα οποιοδήποτε πλήκτρο για να συνεχίσεις». Οι λεπτομέρειες λειτουργίας που ακολουθούν εξετάζονται με τη σειρά μάλλον που είναι πιθανότερο να χρησιμοποιηθούν παρά με τη σειρά αριθμού επιλογής.

Δημιουργία αρχείου - Create file (Επιλογή 1).

Όταν δημιουργείται ένα καινούργιο αρχείο θα πρέπει να δοθούν μέσω του πληκτρολόγιου ένας αριθμός κύριων χαρακτηριστικών, ώστε να καθορισθούν οι διαστάσεις του βασικού ορθογωνικού πίνακα. Σ' αυτά θα περιλαμβάνονται ο αριθμός εγγραφών και ο αριθμός πεδίων. Οι λέξεις - επικεφαλίδες κάθε πεδίου είναι επίσης απαραίτητες.

Μέγεθος αρχείου

Η πρώτη υπενθύμιση στην οθόνη είναι «Enter file size (number of records) δηλ. «Δώσε μέγεθος αρχείου (αριθμό εγγραφών)». Δεν ωφελεί να είστε

πολύ φιλόδοξοι όταν εισάγετε αυτόν τον αριθμό, διότι η ποσότητα διαθέσιμης RAM, πέρα και πάνω από εκείνη που καταλαμβάνει το πρόγραμμα και οι buffers, είναι περίπου 32K. Θα δούμε αργότερα, ότι, καθώς εισάγονται εγγραφές στο αρχείο, η οθόνη συνεχώς εμφανίζει την εκάστοτε *κατάσταση (status)* του αρχείου, που περιλαμβάνει τον αριθμό των bytes που είναι ακόμα ελεύθερα στη RAM. Με τη συνεχή χρήση, ο αριθμός των εγγραφών που προστέθηκαν στο αρχείο μπορεί να αυξηθεί σε σημείο που να προσεγγίζεται το όριο του αριθμού των εγγραφών ή ο αριθμός των ελεύθερων bytes να φαίνεται επικίνδυνα χαμηλός. Στο σημείο αυτό είναι δυνατό να χωρίσουμε το αρχείο σε δύο μέρη χρησιμοποιώντας την επιλογή 8, «Create subfile» (Δημιουργία υποαρχείου).

Αριθμός πεδίων

Η επόμενη υπενθύμιση στην οθόνη είναι: «Enter number of fields required (2-15)» (Δώσε αριθμό απαραίτητων πεδίων (2-15)). Ο μέγιστος αριθμός ορίζεται σε 15 και ο ελάχιστος σε 2. Στο πρόγραμμα περιλαμβάνεται ένας έλεγχος για να αποτρέψει την υπέρβαση του ορίου, αν και θα ήταν εύκολο να τροποποιήσουμε το άνω όριο. Θυμηθείτε ότι ο χώρος της RAM που καταλαμβάνεται από το αρχείο εξαρτάται από το γινόμενο του αριθμού των εγγραφών επί τον αριθμό των πεδίων σε κάθε εγγραφή και επί την ποσότητα του κείμενου σε κάθε πεδίο.

Επικεφαλίδες πεδίων

Κάθε πεδίο πρέπει να έχει μία επικεφαλίδα, αλλοιώς τα δεδομένα μέσα στο πεδίο δεν έχουν κανένα νόημα. Για κάθε πεδίο χρειάζεται διαφορετική επικεφαλίδα, έτσι η υπενθύμιση στην οθόνη, ας πούμε για το πεδίο 3, είναι «Enter field heading 3» δηλ. «Δώσε επικεφαλίδα πεδίου 3». Καμμία επικεφαλίδα δεν πρέπει να ξεπερνάει σε μήκος τους 18 χαρακτήρες. Αν δοθούν περισσότεροι από 18, το πρόγραμμα αγνοεί όλους τους επιπλέον χαρακτήρες χωρίς ειδοποίηση. Το όριο των 18 επιβάλλεται λόγω των απαιτήσεων της Επιλογής 4 που επιτρέπει να βλέπουμε ταυτόχρονα μία σελίδα εγγραφών σε μορφή «πλατιού φύλλου». Το όριο των 18 χαρακτήρων ανά πεδίο τον Τύπο 1 (Mode 1) επιτρέπει να εμφανίζονται οριζόντια ταυτόχρονα μόνο δύο απ' τα πεδία (τα υπόλοιπα πεδία βγαίνουν στην οθόνη με τα πλήκτρα ελέγχου του δρομέα). Δεν έχει σημασία εάν οι επικεφαλίδες πεδίου εισάγονται με κεφαλαία ή μικρά γράμματα, επειδή το πρόγραμμα τα μετατρέπει πάντα, έτσι κι αλλοιώς, σε κεφαλαία.

Προσθήκη εγγραφών - Add records (Επιλογή 5)

Τα στοιχεία που εισάχθηκαν προηγούμενα ήταν απλώς τα κύρια χαρακτηριστικά και κανονικά θα παραμείνουν αμετάβλητα κατά την διάρκεια ζω-

ής του αρχείου. Απ' τη στιγμή που αυτά έχουν εισαχθεί, η οθόνη αλλάζει αυτόματα και εμφανίζεται η Επιλογή 5 «Add records» (προσθήκη εγγραφών) έτοιμη να δεχτεί δεδομένα για το καθένα πεδίο. Σα βοήθεια στις εξηγήσεις που ακολουθούν, ας υποθέσουμε ότι το αρχείο έχει τρία πεδία με επικεφαλίδες: Name (όνομα), Telephone (τηλέφωνο) και Age (ηλικία). Οι υπενθυμίσεις του προγράμματος θα ήταν:

NAME - (θα μπορούσατε να δώσετε BREW P)
TELEPHONE - (Θα μπορούσατε να δώσετε 208 4672)
AGE - (θα μπορούσατε να δώσετε 36).

Μη ξεχνάτε ότι, όπως συμβαίνει και με τις επικεφαλίδες πεδίων, τα στοιχεία των δεδομένων δεν πρέπει να υπερβαίνουν τους 18 χαρακτήρες. Αυτός δεν είναι ένας αυστηρός περιορισμός υπό τον όρο ότι το αρχείο έχει σχεδιαστεί με τον περιορισμό αυτό υπ' όψη. Για παράδειγμα, θα ήταν κακός σχεδιασμός να επιγράψουμε ένα πεδίο «Address» (Διεύθυνση), διότι είναι απίθανο μία πλήρης διεύθυνση να μπορέσει να στριμωχτεί σε 18 χαρακτήρες. Ένα πεδίο θα 'πρεπε να ήταν «Number/ street» (Αριθμός/οδός), το επόμενο «Town» (Πόλη), το επόμενο «Country» (Νομός) και το τελευταίο «Post code» (Ταχυδρομικός κώδικας). Εφόσον διατίθενται 15 πεδία ανά εγγραφή, απομένουν άλλα 11 πεδία για άλλα στοιχεία δεδομένων.

Όταν και το τελευταίο δεδομένο έχει εισαχθεί, η οθόνη καθαρίζει έτοιμη να δεχθεί την είσοδο της επόμενης εγγραφής. Αν δεν πρόκεται να προστεθούν άλλες εγγραφές κατά τη διάρκεια της τρέχουσας διαδικασίας πληκτρολόγησης, το menu μπορεί να επανέλθει στην οθόνη πληκτρολογώντας «EXIT». Αν ο αριθμός των εγγραφών φθάσει τον μέγιστο που εκτιμήθηκε κατά την διάρκεια δημιουργίας του αρχείου, ή αν τα ελεύθερα bytes εξαντληθούν, θα εμφανισθεί το μήνυμα «File full» (Αρχείο Πλήρες), και δε θα γίνουν δεκτές επιπλέον εγγραφές.

Εάν ένα αρχείο υπάρχει ήδη στην RAM, διαλέγοντας την Επιλογή 1 θα καταστρεφόταν, έτσι το πρόγραμμα διατυπώνει μία προειδοποίηση για την περίπτωση που η επιλογή έγινε κατά λάθος, οι λεπτομέρειες της οποίας δίνονται αργότερα στην παράγραφο για την επιλογή «Load file» (Φόρτωση αρχείου).

Απεικόνισε αρχείο - Display file (Επιλογή 4)

Αφού το αρχείο έχει μόλις δημιουργηθεί, ή αν πρόσθετες εγγραφές έχουν μόλις εισαχθεί, το επόμενο λογικό βήμα είναι να δούμε πώς φαίνεται στην οθόνη. Υποθέτοντας ότι το αρχείο έχει μόνο τρεις εγγραφές και καθεμία απ' αυτές δίνει Name (όνομα), Telephone (τηλέφωνο) και Age (ηλικία), η οθόνη θα εμφανίζει την εξής εικόνα πάνω κάτω:

Πίεσε το πλήκτρο διαστήματος για το menu	
NAME	TELEPHONE
BLENKINSOP J	666 6778
GUTSWORTHY M	233 6895
GLUGENHEIMER K	233 1212

Το NAME είναι το σταθερό πεδίο - κλειδί, αλλά το δεύτερο πεδίο, το TELEPHONE, είναι μία μεταβλητή (με την έννοια της απεικόνισης στην οθόνη). 'Αλλα πεδία μπορούν να εμφανιστούν κυκλικά στη δεξιά πλευρά της οθόνης με χρήση των πλήκτρων αριστερού ή δεξιού δρομέα. Για παράδειγμα, εάν πιέζαμε το πλήκτρο του δεξιού δρομέα, η εικόνα της οθόνης θα άλλαζε σε:

Πίεσε το πλήκτρο διαστήματος για το menu	
NAME	AGE
BLENKINSOP J	24
GUTSWORTHY M	83
GLUGENHEIMER K	18

Εάν υπάρχουν περισσότερες από 20 εγγραφές στο αρχείο, χρησιμοποιώντας τα πλήκτρα άνω ή κάτω δρομέα, εμφανίζονται στην οθόνη άλλες ομάδες των 20. Αν ξεπερασθούν τα όρια των πεδίων ή των φακέλλων στην οθόνη, παρατηρείται επανεμφάνιση. Δηλαδή, αν υπήρχαν επτά πεδία και προσπαθούσαμε να μετακινηθούμε σε ένα ανύπαρκτο όγδοο πεδίο, η εικόνα θα αναδιπλωνόταν δίνοντας ξανά το δεύτερο πεδίο. Παρόμοιο φαινόμενο συμβαίνει όταν προσπαθούμε να φέρουμε στην οθόνη μία ανύπαρκτη εικοσάδα εγγραφών.

Απεικόνισε ξεχωριστή εγγραφή - Display single record (Επιλογή 10)

Η επιλογή απεικόνισης του αρχείου είναι χρήσιμη για να σχηματίσουμε μία γενική εικόνα ή να παρατηρήσουμε πώς οι τιμές ενός συγκεκριμένου πεδίου ποικίλουν μεταξύ των εγγραφών. Το μάτι μπορεί να τρέξει από την κορυφή ως την βάση ενός πεδίου και να εντοπίσει εύκολα αντικανονικά ή σημαντικά δεδομένα που εισάχθηκαν. Η Επιλογή 10, απ' την άλλη πλευρά, ψάχνει δια μέσου του αρχείου για μία συγκεκριμένη εγγραφή. Απεικονίζει την πλήρη εγγραφή με όλα τα πεδία χωρίς την ανάγκη να καταφύγουμε στα πλήκτρα των δρομέων.

Παίρνοντας το ίδιο παράδειγμα των τριών εγγραφών που είδαμε πριν, και υποθέτοντας ότι ζητάμε τον φάκελλο του GUTSWORTHY M, η έρευνα θα τελείωνε με την εξής απεικόνιση:

Αριθμός τρέχοντος φακέλλου 2	
NAME	GUTSWORTHY M
TELEPHONE	233 6895
AGE	83
Πίεσε οποιοδήποτε πλήκτρο για να συνεχίσεις	

Η έρευνα απαιτεί να εισαχθεί το πεδίο-κλειδί της εγγραφής. Στην περίπτωση του παραδείγματος, η υπενθύμιση θα έλεγε:

«Enter entry under NAME» (Είσαγε δεδομένο κάτω από το NAME).

Η απάντηση σας θα ήταν GUTSWORTHY M. Πάντως το πρόγραμμα είναι επαρκώς φιλικό προς τον χρήστη για να αντιληφθεί ότι μπορεί να μη θυμάστε την ακριβή ορθογραφία κάθε ονόματος. Το πρόγραμμα θα κάνει την έρευνα αν εισάγετε σωστά μερικά από τα πρώτα γράμματα ή έστω μόνο το πρώτο γράμμα. Για παράδειγμα, εισάγοντας GUT, GU ή έστω G θα αρχίσει η διαδικασία έρευνας. Η έρευνα τελειώνει μόλις βρεθεί η πρώτη εγγραφή που έχει πεδίο-κλειδί που αρχίζει με τα ίδια λίγα γράμματα που έχετε εισάγει. Πάντως, όσο λιγότερα γράμματα εισάγετε, τόσο μεγαλύτερη θα είναι η πιθανότητα να εμφανισθεί ένας μη επιθυμητός φάκελλος ο οποίος, από σύμπτωση, αρχίζει με τα ίδια γράμματα. Αν συμβεί αυτό, μπορείτε να προσθέσετε ακόμη ένα γράμμα —εάν το ξέρετε— και να ξεκινήσει μία νέα έρευνα. Αν αποτύχουν όλ' αυτά, η τελευταία προσπάθεια θα ήταν να μεταφερθείτε στην επιλογή Display File, να διατρέξετε με το βλέμμα τα πεδία-κλειδιά μέχρι να βρεθεί το πλήρες όνομα και κατόπιν να επιστρέψετε στην Επιλογή 10 για μία νέα έρευνα. Αντιλαμβανόμαστε ότι όσο ισχύει το παράδειγμα μας των τριών φακέλλων, τέτοιες λεπτομερείς συμβουλές θα μπορούσαν να φαί-

νονται παράλογες αλλά, όταν ένα αρχείο προσεγγίζει τις εκατό εγγραφές, ίσως να εκτιμηθούν οι συμβουλές. Όταν μία έρευνα διαμέσου του αρχείου αδυνατεί να βρει μία εγγραφή που αντιστοιχεί σε κάποια γράμματα που δόθηκαν για την έρευνα, εμφανίζεται το μήνυμα «No such record on file» (Δεν υπάρχει τέτοιος φάκελλος στο αρχείο).

Ασφάλισε το αρχείο - Save file (Επιλογή 3)

Απ' τη στιγμή που έχει απεικονισθεί το αρχείο, το πιθανότερο είναι να επιθυμείτε να ασφαλίσετε ένα αντίγραφο στην κασέτα ή στο δίσκο. Η πρώτη υπενθύμιση είναι: «Enter file name» (Είσαγε όνομα αρχείου). Επιτρέπεται ένα μέγιστο 8 χαρακτήρων για το όνομα του αρχείου. Αν τυπωθούν περισσότεροι από 8, εμφανίζεται το μήνυμα «Too long: Enter again» (Πολύ μακρύ: Είσαγε ξανά). Οι συνηθισμένες οδηγίες για τη διενέργεια της μεταφοράς στην κασέτα ή δισκέτα θα εμφανισθούν μόλις ληφθεί ένα αποδεκτό όνομα αρχείου.

Φόρτωσε το αρχείο - Load File (Επιλογή 2)

Επειδή η φόρτωση ενός αρχείου θα γραφτεί αυτόματα επάνω και επομένως θα καταστρέψει οποιοδήποτε ήδη υπάρχον αρχείο, έχουν ληφθεί από το πρόγραμμα προφυλάξεις για να διασφαλίσουν ένα υπάρχον αρχείο, εφόσον αυτή η επιλογή διαλέχθηκε κατά λάθος. Αν υπάρχει ήδη κάποιο αρχείο, διαλέγοντας την Επιλογή 2 (ή την Επιλογή 1) εμφανίζεται η ακόλουθη προειδοποίηση συνοδευόμενη από ένα χαρακτηριστικό ηχητικό σήμα:

Αυτή η εκλογή καταστρέφει
υπάρχον αρχείο
Πίεσε οποιοδήποτε πλήκτρο
για να την ματαιώσεις
Αλλιώς πίεσε μαζί τα πλήκτρα
CTRL και M και ξαναδιάλεξε

Συνοψίζοντας, αν η Επιλογή 2 εκλεγεί κατά λάθος, πατώντας οποιοδήποτε πλήκτρο επαναφέρετε το menu χωρίς καμμία ζημιά στο υπάρχον αρχείο. Αν η εκλογή ήταν σκόπιμη, τα πλήκτρα CTRL και M πρέπει να πιεσθούν μαζί, ώστε να επαναφέρουν το menu προτού επανεκλέξετε την Επιλογή 2. Αυτή τη φορά δεν θα υπάρξει πια καμιά προειδοποίηση και η υπενθύμιση «Enter file name» (Είσαγε όνομα αρχείου) εμφανίζεται στην οθόνη. Μπορεί να ξεκινήσει η διαδικασία φόρτωσης, υπό τον όρο ότι το αρχείο της κασέτας είναι έτοιμο στη μηχανή.

Τροποποίησε οποιαδήποτε εγγραφή - Modify any record (Επιλογή 6)

Αυτή η επιλογή σας επιτρέπει να τροποποιήσετε οποιδήποτε πεδίο μιας εγγραφής που έχετε διαλέξει. Επίσης επιτρέπει να διαγράψετε τελείως την εγγραφή από το αρχείο. Η εγγραφή που πρόκειται να τροποποιηθεί ανιχνεύεται πρώτα με απάντηση στη υπενθύμιση «Give record under NAME» (Δώσε στοιχείο εγγραφής που υπάγεται στο NAME) —υποθέτοντας, βέβαια, ότι το πεδίο-κλειδί είναι το NAME. Όπως περιγράψαμε προηγουμένως, το πρώτο ή τα δύο πρώτα γράμματα του πρωτεύοντος πεδίου-κλειδιού είναι αρκετά για να ξεκινήσει η έρευνα. Μόλις θα έχει βρεθεί η εγγραφή παρουσιάζεται η ακόλουθη εικόνα:

Τροποποίησε / αντίγραψε γραμμή με τα πλήκτρα βελών / COPY Γράψε KILL για να διαγράψεις το αρχείο Γράψε EXIT για να επαναφέρεις την σελίδα Επιλογών	
NAME TELEPHONE AGE	GUTSWORTHY M 233 6895 82
X	

Ο δρομέας αρχικά θα βρίσκεται στο σημείο που σημαίνεται σαν X. Υποθέστε ότι ο αριθμός του τηλεφώνου είναι λανθασμένος. Ο δρομέας αντιγραφής μετακινείται πρώτα προς τα άνω, πιέζοντας τα πλήκτρα SHIFT και «up arrow» (άνω βέλους), μέχρι το γράμμα T στην αρχή της δεύτερης γραμμής. Τα πλήκτρα SHIFT και COPY χρησιμοποιούνται κατόπιν για να μετακινηθεί ο δρομέας κατά μήκος της οθόνης μέχρι να φτάσει στο λανθασμένο ψηφίο. Τότε μπορεί να διορθωθεί με τον κανονικό τρόπο. Η διορθωμένη μορφή αναπαράγεται επίσης στη γραμμή της βάσης κατά τη διάρκεια της διαδικασίας διόρθωσης του κειμένου. Μόλις πιέσετε το πλήκτρο ENTER, προβάλλεται η διορθωμένη μορφή. Αν είστε ικανοποιημένοι τελικά, γράψτε EXIT για να επαναφέρετε τις επιλογές του menu.

Προκειμένου να διαγράψετε μία εγγραφή από τα αρχεία, όταν απεικονιστεί η εγγραφή που επιθυμείτε γράψτε KILL. Το κενό στην διάταξη κλείνεται εντελώς και επιβάλλεται απομάκρυνση των άχρηστων στοιχείων.

Δημιούργησε υπο-αρχείο (ερεύνησε οποιοδήποτε πεδίο) -Create subfile (search any field): Επιλογή 8

Εκλέγοντας την Επιλογή 8, εμφανίζεται ένα εναλλακτικό menu. Το πρώτο μισό του menu προσφέρει πέντε τρόπους για να διαλέξετε ποιές εγγραφές από το κύριο αρχείο πρόκειται να διαπραγματευθεί το πρόγραμμα σαν«υποαρχείο». Η δημιουργία ενός υποαρχείου από εγγραφές που μοιράζονται μία ή περισσότερες κοινές ιδιότητες είναι ένα ισχυρό χαρακτηριστικό γνώρισμα του προγράμματος, αν και αυτό, από μόνο του, δεν είναι κριτήριο για να προάγουμε το σύστημα αρχειοθέτησης στην κατάσταση μιας «database». Στο σημείο αυτό, αξίζει να κάνουμε μία μικρή παρέμβαση, για να διασαφηνίσουμε την διάκριση μεταξύ των δύο.

Οι όροι «data file» (αρχείο δεδομένων) και «database» (βάση δεδομένων) συχνά χρησιμοποιούνται σαν να πρόκειται για το ίδιο πράγμα. Αυτό δεν είναι ακριβώς σωστό. Στην πραγματικότητα υπάρχουν βαθειές διαφορές μεταξύ ενός data file και μιας database. Για να αποφύγουμε οποιαδήποτε πιθανή σύγχυση, θα πρέπει να θεωρούμε μία database σαν μία *ολοκληρωμένη συλλογή από εγγραφές,* που η καθεμιά τους περιέχει πληροφορίες για τη *σχέση* μεταξύ τους. Αυτές οι εγγραφές θα μπορούσαν πολύ καλά να κατανεμηθούν σε έναν αριθμό από *διαφορετικά* αρχεία. Αντίθετα, ένα απλό αρχείο περιέχει εγγραφές που είναι όλες του ίδιου τύπου. Μία τυπική database, έχει τα ακόλουθα χαρακτηριστικά:

(α) Μια συλλογή από εγγραφές *σε διασταυρούμενη σχέση,* γνωστή σαν «database» που διατηρούνται ίσως σ' ένα πλήθος από αρχεία.

(β) Οι διασταυρωμένες σχέσεις αυτές είναι αποθηκευμένες με έναν οργανωμένο τρόπο μέσα στην ίδια την database, έτσι ώστε να μπορούν να εκτελεστούν ταχύτατα οι απαιτήσεις για περίπλοκες έρευνες.

(γ) Μία database έρχεται σε επαφή με ένα πρόγραμμα εφαρμογών μέσω ενός κομματιού του software που λέγεται *σύστημα διεύθυνσης της database (database management system)* (DBMS). Έτσι το πρόγραμμα εφαρμογών είναι ανεξάρτητο από αλλαγές στη δομή των δεδομένων.

Είναι φανερό από τις ιδιότητες μιας αληθινής database ότι η Επιλογή 8 δεν έχει αξιώσεις προς αυτή την κατεύθυνση. Πάντως, είναι βέβαια ικανή να διενεργεί γρήγορες ερευνητικές λειτουργίες μέσα στα όρια ενός απλού data file. Θα το ελέγξουμε αυτό περιγράφοντας, πώς ένα υποαρχείο μπορεί να περισταλεί προοδευτικά μέχρι να «επιζήσει» μία συγκεκριμένη εγγραφή από την τελική «σύμπτυξη».

Υποθέστε ότι το αρχείο που φορτώθηκε από την αποθήκη ονομάζεται EUROPE και περιέχει τέσσερα πεδία - Country (χώρα), Area (square miles), (Έκταση (σε τετραγ. μίλια)), Population (πληθυσμός) και Capital (πρωτεύουσα). Επιθυμούμε να βρούμε μία χώρα με έκταση *μικρότερη από* 40000 τετραγ. μίλια με πληθυσμό *μεγαλύτερο ή ίσο από* 500000 κατοίκους. Το έργο μας ξεκινά εκλέγοντας την επιλογή 8 η οποία φέρνει στην οθόνη το ακόλουθο menu έρευνας:

ΑΡΧΕΙΟ (ΔΙΑΔΟΧΙΚΗΣ) ΣΥΜΠΤΥΞΗΣ
(Μη καταστρεπτικό)

Ερεύνησε κάποιο πεδίο για:

(1) < > a
(2) = a
(3) > = a
(4) <=a
(5) Ομάδα χαρακτήρων

Περαιτέρω επιλογές του υποαρχείου:

(6) Ασφάλισε το υποαρχείο
(7) Απεικόνισε το υποαρχείο
(8) Κατάταξε το υποαρχείο ως προς κάποιο πεδίο
(9) Διάγραψε το υποαρχείο από το κύριο αρχείο
(10) Άθροισε στήλη
(11) Τύπωσε το υποαρχείο
(12) Επέστρεψε στο κύριο menu

Διάλεξε μια επιλογή

Τα βήματα από 'δω και πέρα μπορούν να ποικίλουν, αλλά η ακόλουθη προσέγγιση είναι λογική:

(1) Διαλέξτε την Επιλογή 4 (έρευνα για< α). Η πρώτη υπενθύμιση είναι:

Operate on which field?
(Σε ποιό πεδίο να ενεργήσω;)

Η απάντηση θα πρέπει να είναι AREA (έκταση).

(2) Η επόμενη υπενθύμιση θα είναι:

Search to operate on AREA (Πρόκειται να διενεργήσω έρευνα στο πεδίο AREA).
Give search datum/entity (Δώστε δεδομένο / στοιχείο για την έρευνα).

Η απάντηση θα πρέπει να είναι 40000.

Διαλέγοντας την επιλογή 7 (Απεικόνισε το υποαρχείο), απεικονίζεται

το υποαρχείο με ολες τις χώρες που έχουν εκτάσεις μικρότερες από 40.000 τετρ. μίλια. Το menu έρευνας επαναφέρεται τώρα (πιέζοντας το πλήκτρο διαστήματος). Τώρα περιστέλλουμε το υποαρχείο αυτό ακόμη περισσότερο.

(3) Διαλέξτε την Επιλογή 3 (έρευνα για > =α)
Η πρώτη υπενθύμιση είναι:

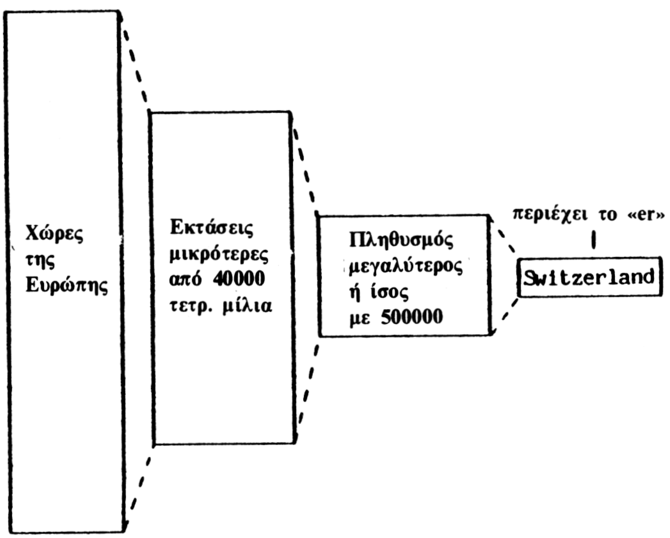
Operate on which field? (Σε ποιό πεδίο να ενεργήσω;)

Η απάντηση θα πρέπει να είναι Population (Πληθυσμός).

(4) Η επόμενη θα είναι:

Search to operate on Population (Πρόκειται να διενεργήσω έρευνα στο πεδίο Population).
Give search datum / entity (Δώστε δεδομένο / στοιχείο για την έρευνα)

Η απάντηση πρέπει να είναι 500000.



Σχ. 4.1. Σύμπτυξη ενός κύριου αρχείου για τις Ευρωπαϊκές χώρες.

Διαλέγοντας την Επιλογή 7 ('Αποψη του υποαρχείου), απεικονίζεται το νέο υποαρχείο με όλες τις χώρες που έχουν έκταση μικρότερη από 40000 τετραγ. μίλια και με πληθυσμό μεγαλύτερο ή ίσο από 500000 κατοίκους. Θα μπορούσαμε, βέβαια, ακόμη περισσότερο χρησιμοποιώντας την Επιλογή 5 (ομάδα χαρακτήρων) και να επιμείνουμε ότι η χώρα πρέπει να περιλαμβάνει το υποαλφαριθμητικό «er». Σύφμωνα με το δικό μας αρχείο για την Ευρώπη (του οποίου τα στοιχεία έχουν συλλεγεί με τη βοήθεια του *Whitaker' s Alma-*

nac,) η τελική σύμπτυξη κατέδειξε μία χώρα μόνο: Switzerland (Ελβετία). Το παράδειγμα εικονογραφείται παραστατικά στο σχήμα 4.1.

Το υπόλοιπο μισό του menu έρευνας προσφέρει μερικές άμεσες επιλογές, συμπεριλαμβανομένης της ασφάλισης ή της κατάταξης του υποαρχείου σε οποιοδήποτε στάδιο. Αυτές χρησιμοποιούνται κατά τον ίδιο ακριβώς τρόπο όπως οι πανομοιότυπες στο κύριο menu. Ωστόσο, αξίζει να επισημάνουμε ότι η δυνατότητα της διάσπασης του κύριου αρχείου, χρησιμοποιώντας την Επιλογή 9 (Διάγραψε το υποαρχείο από το κύριο αρχείο), είναι πολύ χρήσιμη, όταν το κύριο αρχείο αυξάνεται σε υπερβολικό μέγεθος για την RAM. Για παράδειγμα, μπορούμε να δημιουργήσουμε ένα υποαρχείο όλων των ονομάτων τα οποία αρχίζουν με γράμματα που βρίσκονται πριν από το «L». Το υποαρχείο, αφού ασφαλισθεί με ένα άλλο όνομα αρχείου, μπορεί κατόπιν να διαγραφεί από το κύριο αρχείο. Έτσι προκύπτει ένα νέο «κύριο αρχείο» και το αρχικό αρχείο, πολύ ελαφρωμένο πια, μπορεί να ασφαλισθεί ξεχωριστά. Σε σχέση με την παραπάνω επισήμανση «γράμματα που βρίσκονται πριν από το L», θα πρέπει να δοθεί η εξήγηση, ότι οποιαδήποτε επιλογή έρευνας στο menu του υποαρχείου ενεργεί σύμφωνα με τους κωδικούς του ASCII, όταν το πεδίο είναι αλφαβητικό. Έτσι αν χρησιμοποιήσουμε τον τελεστή «< » σε ένα αλφαβητικό πεδίο, υπαγόμενο στο NAME, θα συγκρίνει διαδοχικούς χαρακτήρες του ASCII, όσους χρειάζονται για την απόφαση.

Κατάταξε - Sort (Επιλογή 7)

Η επιλογή 7 στο κύριο menu (ή η 8 στο menu του υποαρχείου) θα κατατάξει ένα αρχείο κατά μία οποιαδήποτε επικεφαλίδα πεδίου. Στο κεφάλαιο 5 διαπραγματευόμαστε λεπτομερώς το θέμα της κατάταξης. Στο μεταξύ ας πούμε, ότι πρόκειται απλώς για μία υπόθεση απάντησης στην υπενθύμιση «Operate on which field?» (Σε ποιό πεδίο να ενεργήσω;). Εάν απαντήσουμε NAME, θα τοποθετήσει όλες τις εγγραφές σε αλφαβητική σειρά κατά το όνομα. Απ' την άλλη μεριά, μία κατάταξη σύμφωνα με την έκταση (AREA), θα είναι αριθμητική παρότι οι αριθμοί βρίσκονται ακόμη σε αλφαριθμητική μορφή. Εξαιτίας αυτού, είναι σημαντικό ότι, όταν δημιουργούμε ένα αρχείο, όλα τα αριθμητικά δεδομένα που εισάγουμε πρέπει να έχουν τον *ίδιο συνολικό αριθμό ψηφίων.* Οι μικροί αριθμοί πρέπει να έχουν αρχικά μηδενικά σαν πρόσθετο περιτύλιγμα. Αν δε γίνει αυτό, δεν πρόκειται να διενεργηθούν σωστά οι αριθμητικές κατατάξεις. Για να δείτε γιατί συμβαίνει αυτό, εξετάστε την ακόλουθη σειρά αριθμών:

```
   45
  763
15777
   94
```

Αν αυτοί είναι αποθηκευμένοι σε αλφαριθμητική μορφή, η κατάταξη διενεργείται από τον κωδικό του ASCII από τον πιο σημαντικό χαρακτήρα προς τον λιγότερο σημαντικό. Στο παραπάνω παράδειγμα, οι πρώτοι χαρακτήρες που εξετάζονται θα είναι οι κωδικοί για τους 4, 7, 1, 9 διότι αυτοί παίρνουν το προβάδισμα από τους λιγότερο σημαντικούς χαρακτήρες. Συνεπώς, αν οι αριθμοί επρόκειται να καταταγούν από τον μεγαλύτερο προς τον μικρότερο, ο υπολογιστής θα έκρινε ότι σωστή είναι η εξής διάταξη:

```
   94
  763
   45
15777
```

Αν οι αριθμοί είχαν από την αρχή εισαχθεί με τα αναγκαία και επαρκή αρχικά μηδενικά, ώστε να γίνουν όλοι οι αριθμοί μεγέθους 5 χαρακτήρων, η κατάταξη θα απέφερε το εξής σωστό αποτέλεσμα:

```
15777
00763
00094
00045
```

Η κατάταξη δεδομένων είναι μία περίπλοκη υπόθεση· μπορεί να είναι απαράδεκτη η χρονοτριβή, αν δεν χρησιμοποιηθούν επαρκείς τεχνικές προγραμματισμού για τον χειρισμό μεγάλων αρχείων, έτσι το επόμενο κεφάλαιο είναι αφιερωμένο αποκλειστικώς σ' αυτό το θέμα.

Τύπωσε το αρχείο - Print file (Επιλογή 9)

Αυτή η επιλογή επιτρέπει την εκτύπωση σε αντίγραφο ενός αρχείου. Σοβαρή χρήση ενός συστήματος αρχειοθέτησης συχνά απαιτεί *απεικονίσεις σε αντίγραφο (hard copy)* εκτός από την οθόνη, έτσι η Επιλογή 9 αξίζει ειδική προσοχή. Κανονικά, η εντολή PRINT υπονοεί «εκτύπωση» στην οθόνη λόγω του ότι εκ κατασκευής έτσι ερμηνεύει ο υπολογιστής την εντολή PRINT. Οι γραμμές εξόδου των περιφερειακών του Amstrad αναγνωρίζονται από τον χαρακτήρα «#» που ακολουθείται από έναν αριθμό. Η παράλληλη έξοδος του εκτυπωτή αναγνωρίζεται από τον αριθμό «8», έτσι οι εντολές εκτύπωσης, που προορίζονται για παραγωγή αντιγράφων, πρέπει να είναι της μορφής:

```
PRINT #8, μεταβλητές
```

Αν και ο Amastrad DMP-1 και DMP-2000 είναι ο επίσημος εκτυπωτής της Amstrad και θα δουλέψει χωρίς πρόβλημα μόλις συνδεθεί, θα υπάρχουν πολλοί τελικοί χρήστες που έχουν ήδη έναν πάρα πολύ καλό εκτυπωτή και

θα επαναστατούσαν στην σκέψη να αγοράσουν κι' άλλον. Δυστυχώς υπάρχει έλλειψη τυποποίησης τόσο στα interfaces των εκτυπωτών, όσο και στους χαρακτήρες ελέγχου που αναγνωρίζονται από το software των εκτυπωτών. Πολλοί πιστεύουν ότι οποιοσδήποτε υπολογιστής με μία παράλληλη (Centronics) υποδοχή στην πίσω πλευρά θα ταιριάζει πάντοτε επιτυχημένα με οποιονδήποτε εκτυπωτή που επαίρεται για το παράλληλο interface του. Δυστυχώς, αυτή η ιδεώδης κατάσταση, σπάνια πραγματοποιείται στην πράξη. Ακόμη και άν τα βύσματα και οι υποδοχές συμβεί να ταιριάζουν (και πολύ συχνά δεν συμβαίνει) μπορεί πάλι να υπάρχει πρόβλημα με τους αντίστοιχους αριθμούς των ακίδων τους. Για παράδειγμα μερικοί υπολογιστές αποστέλλουν ένα carriage return στον εκτυπωτή, αλλά, επειδή και ο εκτυπωτής επίσης παράγει εσωτερικά ένα carriage return, το αποτέλεσμα είναι ένα ανεπιθύμητο διπλό διάστημα μεταξύ των αριθμών των γραμμών. Αυτό μπορεί συχνά να θεραπευτεί με μελέτη του εγχειριδίου (manual) που παρέχεται μαζί με τον εκτυπωτή και κατάλληλο παίξιμο με τους χαρακτήρες ελέγχου. Μία άλλη πιθανή θεραπεία, αλλά που δεν θα πρέπει να υποθετηθεί χωρίς προηγούμενη εμπειρία, είναι να κοπεί ο ακροδέκτης που συνδέεται με την ακίδα 14 στο επίπεδο καλώδιο —αλλά εδώ πάλι, εξαρτάται από τον συγκεκριμένο εκτυπωτή.

Ενώ ακόμη βρισκόμαστε στο θέμα των εκτυπωτών, αξίζει να θυμηθούμε το παλιό γνωμικό —παίρνεις ό,τι πληρώνεις. Οι εκτυπωτές, ιδιαίτερα αυτοί που ανήκουν στη λεγόμενη «νοήμονα» ποικιλία έχουν πέσει δραματικά στην τιμή, παρά την αυξημένη περιπλοκότητά τους. Είναι πολύ ωραίο να έχεις μία ποικιλία από τυπογραφικά στοιχεία διαφόρων ειδών, μεγεθών και πάχους που σε φέρνουν σε αμηχανία, όπως και υπογραμμίσεις, εκθέτες, δείκτες και σε μερικές περιπτώσεις διάφορα χρώματα μελάνης, με την προϋπόθεση όμως ότι όλα αυτά δεν υπάρχουν σε βάρος της αξιοπιστίας. Αφού περάσει ο αρχικός ενθουσιασμός δοκιμάζοντας όλα αυτά τα ιδιαίτερα χαρακτηριστικά, οι περισσότεροι κατασταλάζουν στον αρχικό τύπο στοιχείων του κατασκευστή γιατί το να αναζητάς τους χαρακτήρες ελέγχου για την παραγωγή ειδικών χαρακτήρων γίνεται πολύ εκτενής διαδικασία. Αν ο εκτυπωτής πρόκειται να δουλεύει σκληρά και συχνά, υπάρχουν τέσσερα ερωτήματα σχετικά με την λειτουργία του, τα οποία υπερτερούν σε σημασία απ' όλα τα υπόλοιπα μαζί:

(a) Πόσο γερό και αξιόπιστο μηχάνημα είναι;
(b) Πόσο εύκολο είναι να αλλάξουμε την κεφαλή εκτύπωσης;
(c) Παίρνει τυποποιημένο χαρτί εκτύπωσης;
(d) Έχει τους κατάλληλους μηχανισμούς τριβής και έλξης του χαρτιού εκτύπωσης, χωρίς να το μασάει κάθε τόσο;

Άθροισε οποιαδήποτε στήλη - Totalise any column (επιλογή 11)

Η ικανότητα να αθροίζει το πρόγραμμα μία στήλη έχει έννοια μόνο αν η στήλη είναι εξ ολοκλήρου *αριθμητική* στους χαρακτήρες. Για παράδειγ-

μα, αν η στήλη βρίσκεται κάτω από την επικεφαλίδα πεδίου «Cash in hand (Μετρητά στο χέρι)», τότε η άθροιση είναι μία λογική ενέργεια. Αντίθετα θα ήταν ανοησία να αθροίσει κάποιος μία στήλη με την επικεφαλίδα πεδίου «Name» (΄Ονομα). Η πρώτη υπενθύμιση είναι «Operate on which field?» (Σε ποιό πεδίο να ενεργήσω;). Υποθέτοντας ότι η απάντηση είναι αποδεκτή και ότι υπάρχει το πεδίο, το αποτέλεσμα εμφανίζεται ως εξής:

```
Σύνολο στήλης = ...
Μέσος όρος στήλης = ...
```

Τερμάτισε το πρόγραμμα - End Program (Επιλογή 12)

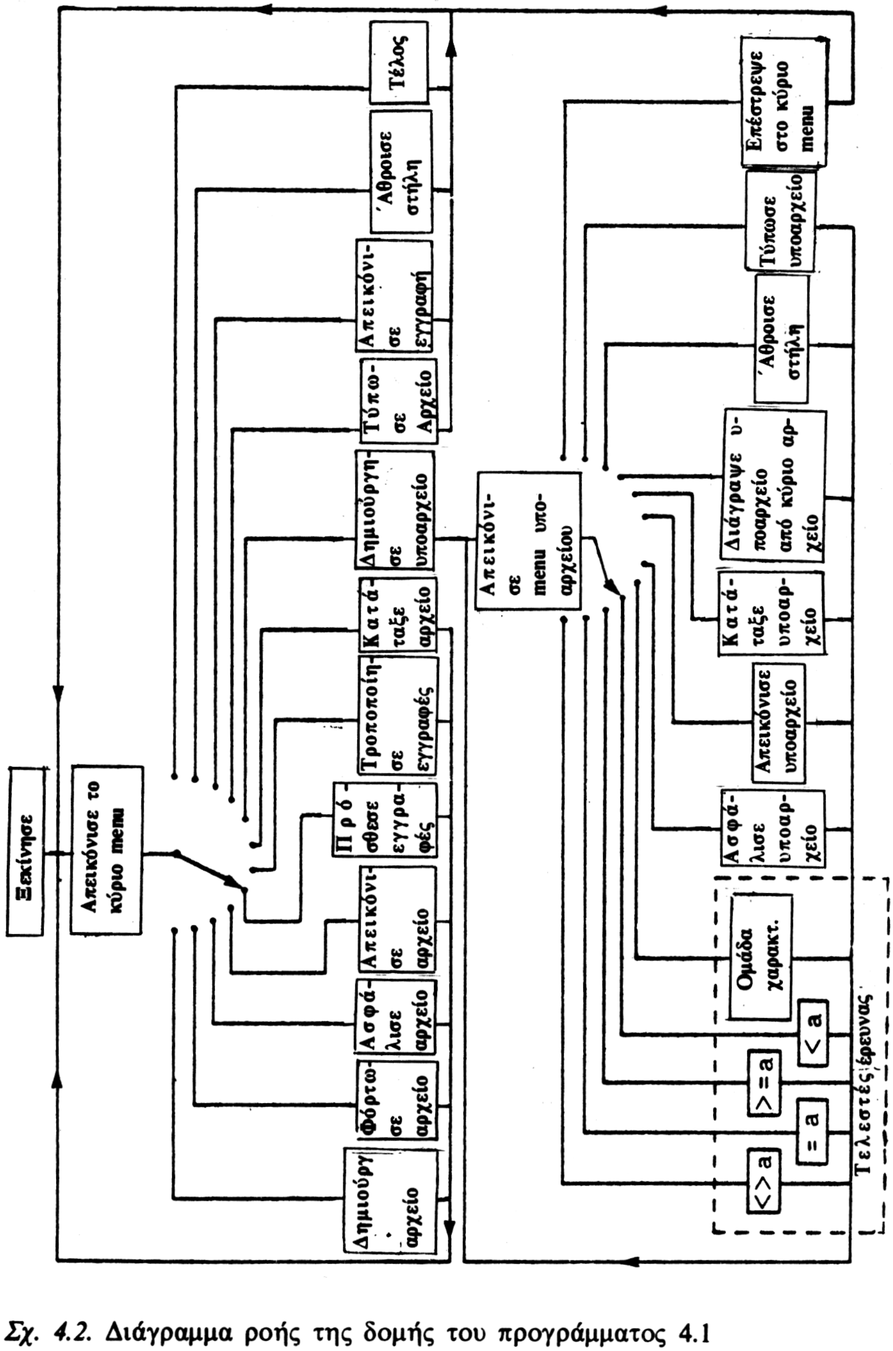
Αυτή η επιλογή απλώς τερματίζει το πρόγραμμα και θέτει την ταχύτητα εγγραφής της κασέτας στην αρχική της τιμή. Δυστυχώς, η HIMEM δεν μπορεί να επανατεθεί στην αρχική της τιμή εκτός αν εκτελεστεί η επαναφορά στο hardware. Πάντως αν το επόμενο πρόγραμμα που θα φορτωθεί δεν καταλαμβάνει πολύ χώρο στη μνήμη, η προηγούμενη αδυναμία που αναφέρθηκε δεν αποτελεί πρόβλημα καθόλου.

Ανάλυση του Προγράμματος

Η λίστα του Προγράμματος 4.1. δείχνει ότι το πραγματικό «πρόγραμμα» καταλαμβάνει λιγότερο από 40 γραμμές και εκτείνεται μόνο μέχρι το END στη γραμμή 360. Το υπόλοιπο της λίστας είναι μία συλλογή από υπορουτίνες. Το σχήμα 4.2 δείχνει τη συνολική στρατηγική, ιδιαίτερα τη σχέση μεταξύ των επιλογών του υποαρχείου και του κυρίου αρχείου.

Ονόματα μεταβλητών που χρησιμοποιούνται στο Πρόγραμμα 4.1

mainsize%= τρέχων αριθμός εγγραφών στο κύριο αρχείο.
subsize%= τρέχων αριθμός εγγραφών στο υποαρχείο.
L%= προσωρινή μεταβλητή προσαρμοσμένη στον τρέχοντα αριθμό, εγγραφών σ' ένα από τα δύο αρχεία ανάλογα με την περίπτωση.
flag%= κατάσταση (status) του αρχείου .1= υπάρχει αρχείο στην RAM, 0= δεν υπάρχει αρχείο στην RAM
filename$= όνομα του αρχείου.
filesize%= μέγιστος αριθμός εγγραφών που μπορεί να υπάρχουν στο κύριο αρχείο
fields%= αριθμός των πεδίων στο αρχείο.
K$= σφαιρική μεταβλητή για γενική χρήση.
total= άθροισμα στήλης.
F%= αριθμός τρέχοντος πεδίου



Σχ. 4.2. Διάγραμμα ροής της δομής του προγράμματος 4.1

R%= αριθμός τρέχουσας εγγραφής
A$ (F%, R%)= δισδιάστατο στοιχείο του πίνακα του αρχείου (ένα πεδίο).
A$ (F%)= επικεφαλίδα του πεδίου F %
SEL%= αριθμός της διαλεγμένης επιλογής
bottom%= βάση της απεικόνισης
top%= κορυφή της απεικόνισης.

Ανακλήσεις υπορουτινών

Όλες οι υπορουτίνες δέχονται έναν τίτλο μέσω εντολών της REM, έτσι εύκολα τοποθετούνται στη λίστα. Σύμφωνα με καθιερωμένη πρακτική, δεν υπάρχουν GOTO προς γραμμές της REM, έτσι αυτές μπορούν να παραλειφθούν από τη λίστα αν επιθυμείτε να εξοικονομήσετε μνήμη. Μέσα στο σύνολο των υπορουτινών οι REM έχουν χρησιμοποιηθεί με φειδώ. Πάρα πολλές REM σε μία λίστα και συνεχείς αναφορές σε αριθμούς γραμμών στις περιγραφές που τις συνοδεύουν μπορεί συχνά να αποβούν από μόνες τους αποτυχία. Πάντως μπορείτε ν' αναγνωρίσετε ότι οι περισσότερες υπορουτίνες ανήκουν στην τυποποιημένη λίστα μας των δομικών που δόθηκε στο προηγούμενο κεφάλαιο.

Κύριες υπορουτίνες	Υπορουτίνες κατώτερου επίπεδου
Κύριο menu:	
Δημιούργησε αρχείο	Βάλε τα στοιχεία που δίνονται ανά γραμμή Belt and braces Πρόσθεσε εγγραφές
Φόρτωσε το αρχείο	Βάλε το όνομα του αρχείου
Ασφάλισε το αρχείο	Βάλε το όνομα του αρχείου
Απεικόνισε το αρχείο	Σχεδίασε γραμμή
Πρόσθεσε εγγραφές	Πίεσε κάποιο πλήκτρο Βάλε τα στοιχεία που δίνονται ανά γραμμή
Τροποποίησε κάποια εγγραφή	Βρες την εγγραφή Σχεδίασε γραμμή
Κατάταξε ως προς κάποιο πεδίο	Κατάταξε το αρχείο Βρες την επικεφαλίδα του πεδίου
Τύπωσε το αρχείο	(δεν υπάρχουν υπορουτίνες κατώτερου επίπεδου)
Απεικόνισε εγγραφή	Βρες την εγγραφή Σχεδίασε γραμμή Πίεσε κάποιο πλήκτρο
Άθροισε στήλη	Βρες την επικεφαλίδα του πεδίου

	Πίεσε κάποιο πλήκτρο
Menu του υποαρχείου:	
Δημιούργησε υποαρχείο	Ερεύνησε το menu του αρχείου
	Δώσε τίτλο
	Ερεύνησε τον κατάλογο
	Βρες την επικεφαλίδα του πεδίου
	Βάλε τα στοιχεία που
	δίνονται ανά γραμμή
	Πίεσε κάποιο πλήκτρο
	Ερεύνησε< >
	Ερεύνησε =
	Ερεύνησε > =
	Ερεύνησε< =
	Ερεύνησε για ομάδα χαρακτήρων
Διάγραψε το υποαρχείο	(δεν υπάρχουν υπορουτίνες
	κατώτερου επίπεδου)

Πίνακας 4.1. Ανακλήσεις υπορουτινών

Οι περισσότερες επιλογές υλοποιούνται από υπορουτίνες πρώτου επιπέδου, οι οποίες μπορούν να καλούν υπορουτίνες δευτέρου επιπέδου που με τη σειρά τους μπορούν ακόμα να καλούν άλλες τρίτου επιπέδου. Ο πίνακας 4.1 δείχνει τις κλήσεις των υπορουτινών και πρέπει να αποβεί χρήσιμος αν αποφασίσετε να ακολουθήσετε τη συμβουλή που δόθηκε πιο πριν, να πληκτρολογήσετε δηλαδή το πρόγραμμα κατά μία ενότητα κάθε φορά.

Χρήση του Προγράμματος 4.1

Τα πλεονεκτήματα των συστημάτων αρχειοθέτησης που βασίζονται στην RAM είναι η ταχύτητα της διαδικασίας και ο βαθμός της πολυσυνθετότητας που μπορούν να επιτευχθούν. Το μειονέκτημα, βέβαια, είναι το όριο που επιβάλλεται από την ανάγκη να φορτωθεί ολόκληρο το αρχείο, ακόμη και αν θέλουμε να παρατηρήσουμε έναν φάκελλο. Ευτυχώς ο Amstrad είναι εφοδιασμένος επαρκώς με RAM έτσι, παρά τον περιορισμό στο μήκος, το πρόγραμμα θα αποδειχθεί ιδιαίτερα πρακτικό και αξίζει σίγουρα τον κόπο της πληκτρολόγησής του.

Η επιλογή κατάταξης του αρχείου (sort file) είναι προφανώς υποψήφια για πειραματισμό. Οι ρουτίνες κατάταξης της BASIC, ακόμη και οι καλύτερες, είναι εξοργιστικά αργές όταν υπάρχει μεγάλος αριθμός εγγραφών. Η ταχύτητα μπορεί να αυξηθεί δραστικά, εάν η μορφή σε BASIC αντικατασταθεί από μία από τις παραλλαγές γλώσσας μηχανής που περιγράφονται στο επόμενο κεφάλαιο.

Περίληψη

'Ολες οι παρακάτω εντολές αναφέρονται στο Πρόγραμμα 4.1.

1. Μόνο οι Επιλογές 1 και 2 μπορούν να χρησιμοποιηθούν, εκτός αν υπάρχει ήδη αρχείο στην RAM.
2. Οι εγγραφές μπορούν να έχουν έως 15 πεδία.
3. Οι επικεφαλίδες των πεδίων περιορίζονται στους 18 χαρακτήρες.
4. Μόνο δύο πεδία είναι ορατά στην οθόνη κάθε φορά, το ένα δε απ' αυτά είναι το πεδίο-κλειδί.
5. 'Ολα τα πεδία που εισάγονται μετατρέπονται σε κεφαλαία.
6. Η εντολή EXIT εισάγεται για να τερματισθεί η εισαγωγή δεδομένων.
7. Κατά την χρησιμοποίηση της Επιλογής 4, τα πλήκτρα αριστερού δεξιού βέλους φέρνουν στην οθόνη άλλα πεδία, τα πλήκτρα άνω / κάτω βέλους φέρνουν στην οθόνη άλλες εικοσάδες εγγραφών, το δε πλήκτρο διαστήματος επαναφέρει το menu.
8. Το όνομα του αρχείου δεν μπορεί να ξεπερνά τους 8 χαρακτήρες.
9. Αν η Επιλογή 1 (Δημιούργησε αρχείο) ή η Επιλογή 2 (Φόρτωσε αρχείο) εκλεγεί κατά λάθος, πατώντας οποιοδήποτε πλήκτρο επαναφέρετε το menu.
10. Αν μία από τις δύο παραπάνω επιλογές εκλεγεί σκόπιμα, πρέπει να την επιλέξετε ξανά, αφού πρώτα πιέσετε τα πλήκτρα CTRL και M μαζί.
11. 'Ενα υποαρχείο μπορεί να αποσπασθεί από το κύριο αρχείο και να ασφαλισθεί. 'Ετσι προκύπτει ένα ξεχωριστό αυτοδύναμο κύριο αρχείο.
12. 'Ενα κύριο αρχείο μπορεί να συντομευθεί αν διαγράψουμε τις εγγραφές του υποαρχείου απ' αυτό και ασφαλιστεί μετά ξεχωριστά.
13. Τα αρχεία μπορούν να καταταγούν με βάση οποιαδήποτε επικεφαλίδα πεδίου.
14. Αριθμητικά πεδία δεν κατατάσσονται σωστά, εκτός αν όλα τα δεδομένα εισαχθούν απ' την αρχή με τον ίδιο αριθμό ψηφίων, χρησιμοποιώντας επομένως αρχικά μηδενικά όπου χρειάζονται.
15. Η επιλογή εκτύπωσης του αρχείου μπορεί να απαιτήσει μια μικρή τροποποίηση ανάλογα με τον τύπο του εκτυπωτή.

Ασκήσεις

'Ολες οι ερωτήσεις αναφέρονται, στη λίστα του Προγράμματος 4.1

4.1. Ποιός είναι ο σκοπός του προγράμματος μεταξύ των γραμμών 20 έως 50;

4.2. SPEED WRITE 0 είναι η γραμμή πριν το END. Γιατί;

4.3. Γιατί χρησιμοποιείται το ERASE A$ στο BELT and BRACES module;

4.4. Η γραμμή 1370 χρησιμοποιεί VPOS (0) - 2. Γιατί;

Κεφάλαιο Πέμπτο

Έρευνα και Ταξινόμηση

Εισαγωγή

Σ' όλο αυτό το τμήμα του βιβλίου θα διαπραγματευθούμε ποιοτικά την έρευνα και την ταξινόμηση, για να αποφύγουμε να κουράσουμε τον αναγνώστη με πάρα πολλή ξερή ακαδημαϊκή ανάλυση. Η επιλογή των αλγορίθμων που διαπραγματευόμαστε είναι ένα απλό δείγμα από πολλούς που έχουν επινοηθεί για εσωτερική ταξινόμηση. Θα προχωρήσουμε προοδευτικά από μία απλή εναλλακτική ταξινόμηση (exchange sort), μέσω της bubble sort, της ταξινόμησης φθίνοντος βήματος (diminishing increment) και της ταχείας ταξινόμησης (quicksort) μέχρι μία πλήρη ταξινόμηση δισδιάστατης αλφαριθμητικής σειράς στοιχείων, γραμμένη σε γλώσσα μηχανής. Η τελευταία είναι ικανή να ταξινομήσει ένα πλήθος εγγραφών αρχείου μέσα σε λίγα δευτερόλεπτα. Τελικά καλύπτουμε και αλγορίθμους για έρευνα σε γειτονικές λίστες (πίνακες στοιχείων), την απλή διαδοχική έρευνα (sequential search) και την αποδοτικότερη δυαδική έρευνα (binary search).

Ταξινόμηση πινάκων

Στις υπορουτίνες Ταξινόμησης που είναι γραμμένες σε BASIC σ' αυτό το κεφάλαιο έχουν δοθεί αριθμοί γραμμών που ξεκινούν από το 1000. Οι προηγούμενες γραμμές, αν υπάρχουν, ασχολούνται με ένα απλό πρόγραμμα ελέγχου που χρησιμοποιείται για να διατάξει την σειρά των στοιχείων κατά ένα ορισμένο τρόπο και να απεικονίσει τον χρόνο εκτέλεσης. Οι χρονικοί πίνακες που δίνονται ενδεχομένως χρειάζονται κάποιες διευκρινήσεις. Κάθε ομάδα δοκιμών εκτελείται για την ακόλουθη σειρά του πίνακα:

(a) Αντίστροφη διάταξη: Ο πίνακας ελέγχου διαμορφώνεται με ακέραιους σε ανάστροφη διάταξη από την απαιτούμενη.

(b) Σχεδόν καταταγμένη: Το τελευταίο στοιχείο στον πίνακα ελέγχου ταξινομείται στο σωστό μέρος στην κεντρική θέση του πίνακα.

(c) Τυχαία διάταξη: Αυτός είναι ένας πίνακας γεμάτος από τυχαίους αριθμούς.

Οι χρόνοι που δίνονται είναι τυπικοί και μπορεί να ποικίλουν ανάλογα

με τους συγκεκριμένους αριθμούς που χρησιμοποιούμε.

Η εναλλακτική ταξινόμηση (exchange sort)

Αυτός είναι ο απλούστερος από όλους τους αλγορίθμους ταξινόμησης αλλά είναι μάλλον αργός. Η τεχνική του, όπως φαίνεται στο Πρόγραμμα 5.1, βασίζεται στη σύγκριση διαδοχικών στοιχείων σε μία μη ταξινομημένη λίστα ή σ' ένα πίνακα. Το θέμα είναι να ταξινομήσουμε έναν πίνακα ακεραίων A% (N) σε αύξουσα διάταξη. Ξεκινώντας από το A% (1), διενεργείται ένα πέρασμα μέσα από τον πίνακα, μέχρι να ευρεθούν δύο διαδοχικά στοιχεία, σε φθίνουσα διάταξη. Τα προβληματικά αυτά στοιχεία του πίνακα εναλλάσσονται όσον αφορά τη θέση και ο κύκλος ξαναρχίζει από το A% (1). Ο κύκλος επαναλαμβάνεται τόσες φορές όσες χρειάζεται μέχρι να διαταχθεί η λίστα τελείως. Η τεχνική βέβαια αποδίδει αλλά δεν χρησιμοποιείται ευρέως. Το πλεονέκτημά της στηρίζεται στην απλότητα της.

```
10 REM INTEGER SORT DEMONSTRATION
20 CLS
30 INPUT"Sort how many integers";number%
40 PRINT
50 REM FILL AND DISPLAY RANDOM ARRAY
60 DIM A%(NUMBER%)
70 FOR N%=1 TO NUMBER%
80 A%(N%)=10000*RND
90 PRINT A%(N%)
100 NEXT
110 PRINT:PRINT
120 PRINT"SORTING ARRAY"
130 PRINT:PRINT
140 START=TIME/300
150 GOSUB 1000
160 T=TIME/300-START
170 FOR N%=1 TO NUMBER%
180 PRINT A%(N%)
190 NEXT
200 PRINT
210 PRINT"RECORDS SORTED=";NUMBER%
220 PRINT
230 PRINT"SORTING TIME=";ROUND(T,2);"SEC
ONDS"
240 END
250 '
260 '
999 REM EXCHANGE SORT SUBROUTINE
```

```
1000 N%=0
1010 WHILE N%<NUMBER%-1
1020 N%=N%+1
1030 IF A%(N%)>A%(N%+1) THEN T%=A%(N%):A
%(N%)=A%(N%+1):A%(N%+1)=T%:N%=0
1040 WEND
1050 RETURN
```

Πρόγραμμα 5.1 Εναλλακτική ταξινόμηση.

Οι σχετικοί χρόνοι για διάφορες διατάξεις φαίνονται στον Πίνακα 5.1.

Πίνακας 5.1. Διάρκειες εναλλακτικής ταξινόμησης.

Διάταξη πίνακα	Στοιχεία	Χρόνος εκτέλεσης
Αντίστροφη	20	13.40 δευτερόλεπτα
Αντίστροφη	50	197.92 δευτερόλεπτα
Σχεδόν καταταγμένη	20	1.36 δευτερόλεπτα
Σχεδόν καταταγμένη	50	8.53 δευτερόλεπτα
Σχεδόν καταταγμένη	00	34.12 δευτερόλεπτα
Τυχαία	20	8.6 δευτερόλεπτα
Τυχαία	50	142.92 δευτερόλεπτα

Η μελέτη του Πίνακα 5.1 δείχνει ότι ο χρόνος ταξινόμησης αυξάνει επικίνδυνα καθώς ο αριθμός των στοιχείων αυξάνεται. Ωστόσο, ο χρόνος που απαιτείται για την εκτέλεση όταν ο πίνακας είναι σχεδόν καταταγμένος είναι λογικός. Ο πίνακας επίσης αποκαλύπτει ότι η χειρότερη περίπτωση σε μία εναλλακτική ταξινόμηση είναι όταν ο πίνακας είναι σε αντίστροφη διάταξη. Αυτό μόνο μπορούμε να προδικάσουμε. Ο αριθμός συγκρίσεων που απαιτείται στη χειρότερη περίπτωση δίνεται από το

$$1/6\ (N^3 + 5N - 6)$$

όπου N είναι ο αριθμός των στοιχείων που πρέπει να ταξινομηθούν. Ο απαιτούμενος χρόνος θα είναι επομένως ανάλογος προς το N^3 αν το N είναι μεγάλο.

Η bubble sort (ταξινόμηση φυσαλλίδας)

Η bubble sort θεωρείται σαν βελτίωση στην εναλλακτική ταξινόμηση γιατί ο χρόνος δεν σπαταλάται συγκρίνοντας διαδοχικά στοιχεία του πίνακα

που είναι ήδη τοποθετημένα σωστά στην τελική τους θέση. Αυτή είναι πιθανόν η πιο δημοφιλής από τις ρουτίνες ταξινόμησης υπό τον όρο ότι υπάρχει μόνο ένας μικρός αριθμός στοιχείων. Επίδειξη μιας ταξινόμησης φυσαλλίδας δίνεται στο Πρόγραμμα 5.2.

Το Πρόγραμμα 5.2 αποτελείται από έναν εσωτερικό και έναν εξωτερικό βρόχο ελέγχου. Όλο και περισσότερα ζεύγη του πίνακα συγκρίνονται και, αν χρειάζεται, εναλάσσονται σε θέση μέσα στον εσωτερικό βρόχο.

```
999 REM BUBBLE SORT ROUTINE
1000 SIZE%=NUMBER%
1010 WHILE SIZE%>1
1020 SIZE%=SIZE%-1
1030 FOR N%=1 TO SIZE%
1040 IF A%(N%)>A%(N%+1) THEN T%=A%(N%):A
%(N%)=A%(N%+1):A%(N%+1)=T%
1050 NEXT
1060 WEND
1070 RETURN
```

Πρόγραμμα 5.2. Υπορουτίνα bubble sort

Ο μεγαλύτερος ακέραιος του πίνακα ανέρχεται πάντοτε σαν «φυσαλλίδα» δια μέσου όλων των στοιχείων μέχρι την τελική θέση. Δεν χρειάζεται πλέον να αναμείξουμε αυτόν τον ακέραιο σε περαιτέρω συγκρίσεις, αφού θα βρίσκεται στην τελική του θέση στον πίνακα. Ο μετρητής του εξωτερικού βρόχου, L%, μπορεί έτσι να ελαττώσει κατά ένα την τιμή του. Στις επόμενες σειρές των κύκλων του εσωτερικού βρόχου, ο επόμενος μεγαλύτερος ακέραιος ανέρχεται σαν φυσαλλίδα δια μέσου της διάταξης των στοιχείων μέχρι τη διπλανή της τελευταίας θέσης στον πίνακα, και ούτω καθ' εξής, μέχρι όλος ο πίνακας να ταξινομηθεί πλήρως. Παρεπιμπτόντως, πρέπει να αναφερθεί ότι ορισμένοι ακριβολόγοι διαφωνούν και ισχυρίζονται ότι ο αλγόριθμος αυτός δεν είναι καθόλου bubble sort. Προτιμούν να ονομάζουν αυτόν «ripple sort» (ταξινόμηση κυματισμού) αφού η μετακίνηση γίνεται σε κατερχόμενη διεύθυνση μάλλον παρά σε ανερχόμενη.

Ο πίνακας 52 δείχνει τα δεδομένα για την διάρκεια εκτέλεσης για το πρόγραμμα bubble sort.

Η μελέτη του Πίνακα 5.2 αποδεικνύει ότι η bubble sort είναι, γενικά, μία βελτίωση στην εναλλακτική ταξινόμηση από πλευράς χρόνου εκτέλεσης. Ο μέσος αριθμός συγκρίσεων δίνεται προσεγγιστικά από το $1/2\ N^2$, όπου N είναι ο αριθμός των στοιχείων του πίνακα. Έτσι θα χρειασθεί τέσσερεις φορές περισσότερος χρόνος για να ταξινομήσουμε διπλάσιο αριθμό στοιχείων. Πάντως η απόδοση, όταν η σειρά των στοιχείων είναι σχεδόν καταταγμένη, είναι οριακά καλύτερη από την εναλλακτική ταξινόμηση. Αυτό μπορεί να βελτιωθεί, χωρίς να απαιτήσει πάρα πολύ πρόσθετο κόπο, εισάγοντας έναν «δείκτη εναλλαγής» (swop flag) όπως φαίνεται στο Πρόγραμμα 5.3. Η διαδικασία επιτρέπει τον τερματισμό της ταξινόμησης νωρίτερα, εάν

Πίνακας 5.2. Διάρκειες της bubble sort.

Διάταξη πίνακα	Στοιχεία	Χρόνος εκτέλεσης
Αντίστροφη	20	2.25 δευτερόλεπτα
Αντίστροφη	50	13.79 δευτερόλεπτα
Αντίστροφη	100	54.83 δευτερόλεπτα
Σχεδόν καταταγμένη	20	1.39 δευτερόλεπτα
Σχεδόν καταταγμένη	50	8.08 δευτερόλεπτα
Σχεδόν καταταγμένη	100	31.54 δευτερόλεπτα
Τυχαία	20	1.74 δευτερόλεπτα
Τυχαία	50	10.80 δευτερόλεπτα
Τυχαία	100	42.13 δευτερόλεπτα

δεν ανιχνευθούν εναλλαγές σε κάποιον κύκλο του εσωτερικού βρόχου. Ο δείκτης εναλλαγής, SF%, μηδενίζεται στην αρχή κάθε εσωτερικού βρόχου και παίρνει την τιμή ένα μέσα στον βρόχο μόνον αν γίνει κάποια εναλλαγή. Μόλις ένας κύκλος τελειώσει με την SF% μηδενική, η σειρά των στοιχείων πρέπει να έχει ταξινομηθεί.

```
998 REM BUBBLE SORT SUBROUTINE
999 REM (WITH SWOP FLAGS)
1000 SIZE%=NUMBER%
1010 SF%=0
1020 SIZE%=SIZE%-1
1030 FOR N%=1 TO SIZE%
1040 IF A%(N%)>A%(N%+1) THEN T%=A%(N%):A
%(N%)=A%(N%+1):A%(N%+1)=T%:SF%=1
1050 NEXT
1060 IF SF%=1 THEN 1010
1070 RETURN
```

Πρόγραμμα 5.3. Bubble sort με δείκτες εναλλαγής

Οι συγκριτικές διάρκειες χρησιμοποιώντας το σύστημα δεικτών φαίνονται στον Πίνακα 5.3.

Όπως μπορεί να φανεί από τον Πίνακα 5.3, η απόδοση στην περίπτωση της σχεδόν καταταγμένης σειράς στοιχείων βελτιώνεται σημαντικά με χρήση των δεικτών εναλλαγής, με αντίτιμο την οριακή αύξηση στον χρόνο εκτέλεσης κάτω από άλλες συνθήκες.

Η ταξινόμηση φθίνοντος βήματος (Diminishing increment sort)

Αυτή η ομάδα αλγορίθμων ταξινόμησης μερικές φορές αποδίδεται στον

Πίνακας 5.3. Συνδυασμός bubble / δείκτη εναλλαγής.

Διάταξη πίνακα	Στοιχεία	Χρόνος εκτέλεσης
Αντίστροφη	20	2.38 δευτερόλεπτα
Αντίστροφη	50	14.50 δευτερόλεπτα
Αντίστροφη	100	57.62 δευτερόλεπτα
Σχεδόν καταταγμένη	20	1.01 δευτερόλεπτα
Σχεδόν καταταγμένη	50	6.12 δευτερόλεπτα
Σχεδόν καταταγμένη	100	24.27 δευτερόλεπτα
Τυχαία	20	1.92 δευτερόλεπτα
Τυχαία	50	10.82 δευτερόλεπτα
Τυχαία	100	44.33 δευτερόλεπτα

D.L. Shell που επινόησε την Shellsort του το 1959. Όμως, υπάρχουν πολυάριθμες ποικιλίες πάνω στον ίδιο βασικό σκελετό. Οι διαφορές είναι μικρές, αλλά μερικές φορές σημαντικές, έτσι θα ασχοληθούμε με δύο μόνο τέτοιες παραλλαγές. Αν και οι ρουτίνες της bubble sort είναι ικανοποιητικές για ένα μικρό αριθμό στοιχείων, ο χρόνος εκτέλεσης αυξάνεται επικίνδυνα όταν υπερβούμε πάνω - κάτω τα 15. Για να δείτε την αργοπορία προκειμένου για πολλούς αριθμούς, προσπαθείστε να τρέξετε το Πρόγραμμα 5.3 με 1000 ακέραιους. Η σειρά θα ταξινομηθεί περίπου μία μέρα αργότερα! Μία βελτίωση; όπου αναμειγνύονται μεγάλες τιμές του N, είναι να χρησιμοποιήσουμε μία από τις πολλές παραλλαγές ταξινόμησης φθίνοντος βήματος, απ' τις οποίες το πρόγραμμα 5.4 είναι ίσως το απλούστερο παράδειγμα.

Η ταχύτητα εκτελέσεως της bubble sort για μεγάλες λίστες περιορίζεται από την ανάγκη να συγκρίνονται μόνο *διαδοχικά* στοιχεία της διάταξης. Αυτό σημαίνει ότι τα στοιχεία μπορούν μόνο να μετακινηθούν κατά μία θέση στον πίνακα κάθε φορά. Η ταξινόμηση φθίνοντος βήματος είναι μία απόπειρα να ξεπερασθεί αυτό το πρόβλημα κάνοντας τις αρχικές συγκρίσεις με στοιχεία που είναι τοποθετημένα μακριά το ένα από το άλλο. Η διαδικασία ταξινόμησης ξεκινά κάνοντας συγκρίσεις και εναλλαγές σε στοιχεία που απέχουν κατ' αυτό το μεγάλο βήμα. Συνακόλουθα, στοιχεία που βρίσκονται πιο κοντά μεταξύ τους ταξινομούνται καθώς το βήμα προοδευτικά μειώνεται σε κάθε κύκλο ταξινόμησης. Μία σημαντική προϋπόθεση είναι ότι το τελικό πέρασμα ταξινόμησης πρέπει να εκτελεσθεί με βήμα ένα. Τα προηγούμενα περάσματα μετακινούν τα στοιχεία πλησιέστερα προς τις τελικές τους θέσεις στον πίνακα. Το τελικό πέρασμα που εκτελεί μία συνηθισμένη bubble sort, εκτελείται πολύ γρήγορα αφού η σειρά των στοιχείων είναι χοντρικά ήδη διαταγμένη.

Αν η προηγούμενη περιγραφή μοιάζει λίγο δύσκολη να κατανοηθεί με την πρώτη προσπάθεια, είναι μερικές φορές χρήσιμο να απεικονίσουμε τον μηχανισμό με άλλο τρόπο. Παρατηρήσαμε νωρίτερα ότι η bubble sort είναι

επαρκής σε ικανοποιητικό βαθμό για ταξινόμηση μικρών αριθμών από στοιχεία. Επίσης σημειώσαμε ότι η χρήση ενός συστήματος δείκτη εναλλαγής επιταχύνει σημαντικά την εκτέλεση μιας bubble sort όταν τα στοιχεία είναι πάνω - κάτω σε τάξη. Η ταξινόμηση φθίνοντος βήματος αξιοποιεί και τα δύο αυτά στοιχεία. Στην ουσία, ο πίνακας διασπάται σε μικρά σύνολα τα οποία ταξινομούνται σύμφωνα με την bubble sort. Αυτά αναμειγνύονται για να σχηματίσουν μεγαλύτερα σύνολα που θα είναι περίπου σε τάξη. Αυτά τα μεγαλύτερα σύνολα θα ταξινομηθούν με την bubble sort και θα αναμειχθούν για να σχηματίσουν ακόμη μεγαλύτερα σύνολα και ούτω καθεξής μέχρι να μείνουμε με μία μεγάλη χονδρικά διαταγμένη λίστα. Αυτή ταξινομείται τελικά με bubble sort και με βάση τις επισημάνσεις που έγιναν πιο πριν, θα είναι αποτελεσματική. Στο Πρόγραμμα 5.4 ο αριθμός των συνόλων προοδευτικά υποδιπλασιάζεται κάθε φορά γύρω από τον εξωτερικό βρόχο, αλλά ο αριθμός των στοιχείων μέσα σ' ένα σύνολο διπλασιάζεται. Ένα από τα καλύτερα κλειδιά για να καταλάβουμε τον μηχανισμό μιας ταξινόμησης φθίνοντος βήματος, ή στην πραγματικότητα οποιουδήποτε άλλου αλγορίθμου, είναι να χρησιμοποιήσουμε έναν διαγνωστικό πίνακα. Η ιδέα είναι να ακολουθήσουμε διεξοδικά το πρόγραμμα στο χαρτί, χρησιμοποιώντας αυθαίρετα δοκιμαστικά δεδομένα και κρατώντας σημειώσεις για τις διάφορες μεταβλητές - κλειδιά όπως μετρητές των βρόχων κ.λπ.

```
998 REM DIMINISHING INCREMENT SORT
999 REM (VERSION 1)
1000 E%=INT(LOG(NUMBER%)/LOG(2))
1010 F%=2^E%
1020 FOR G%=1 TO E%
1030 F%=F%/2:M%=NUMBER%-F%
1040 SF%=0
1050 FOR N%=1 TO M%
1060 IF A%(N%)>A%(N%+F%) THEN T%=A%(N%):
A%(N%)=A%(N%+F%):A%(N%+F%)=T%:SF%=1
1070 NEXT
1080 M%=M%-1
1090 IF SF%=1 THEN 1040
1100 NEXT
1110 RETURN
```

Πρόγραμμα 5.4. Ταξινομηση φθίνοντος βηματος (Παραλλαγή 1).

Οι διάρκειες για την απλή ταξινόμηση φθίνοντος βήματος δίνονται στον Πίνακα 5.4 που ακολουθεί.

Μερικά ενδιαφέροντα στοιχεία αποκαλύπτονται από τον Πίνακα 5.4. Το πιο κτυπητό είναι η αύξηση στην ταχύτητα, λόγω της περισσότερο γραμμικής σχέσης μεταξύ χρόνου εκτέλεσης και του αριθμού των στοιχείων του πίνακα. Προσέξτε ότι γι' αυτόν τον αλγόριθμο, η χειρότερη περίπτωση εφαρμογής του δεν είναι πια η κατάσταση αντίστροφης διάταξης. Η τιμή αυτή έχει μεταβιβασθεί στους τυχαία φτιαγμένους πίνακες.

Πίνακας 5.4. Διάρκειες ταξινόμησης φθίνοντος βήματος (παραλλαγή 1).

Διάταξη πίνακα	Στοιχεία	Χρόνος εκτέλεσης
Αντίστροφη	100	8.96 δευτερόλεπτα
Αντίστροφη	200	20.96 δευτερόλεπτα
Αντίστροφη	300	31.64 δευτερόλεπτα
Σχεδόν καταταγμένη	100	7.54 δευτερόλεπτα
Σχεδόν καταταγμένη	200	17.57 δευτερόλεπτα
Σχεδόν καταταγμένη	300	29.37 δευτερόλεπτα
Τυχαία	100	21.65 δευτερόλεπτα
Τυχαία	200	71.19 δευτερόλεπτα
Τυχαία	300	136.28 δευτερόλεπτα

Το Πρόγραμμα 5.5 είναι μία παραλλαγή της απλής ταξινόμησης ελαττωμένου βήματος. Η βασική διαφορά είναι στον τρόπο που τα σύνολα υποδιαιρούνται και συντίθενται. Η εκλογή των βημάτων είναι καθαρά αυθαίρετη, με τον όρο το τελικό βήμα να είναι 1. Διαφορετικές τιμές του βήματος μπορεί να δουλέψουν εξίσου καλά, χειρότερα, ή ακόμη καλύτερα. Προσέξτε ότι μία σταθερή (2) προστίθεται στη γραμμή 1020. Βρήκαμε την απόδοση ελαφρά βελτιωμένη μ' αυτήν την πρόσθεση για μεγάλους αριθμούς στοιχείων της διάταξης. Ίσως βοηθά τα σύνολα να αναμειγνύονται καλύτερα. Η ανάλυση της Shellsort είναι πολύ δύσκολη και εκτιμήσεις του αριθμού των συγκρίσεων που απαιτούνται, έχουν σήμερα υπολογισθεί μόνο για ειδικές συνθήκες. Γενικά, το Πρόγραμμα 5.5. είναι μια βελτίωση του προηγουμένου όσον αφορά τον χρόνο εκτέλεσης. Αυτό είναι φανερό από την σύγκριση των Πινάκων 5.4 και 5.5. Η πιο σημαντική βελτίωση είναι στην ταξινόμηση μιας τυχαίας παραγόμενης διάταξης. Η απόδοση για την σχεδόν καταταγμένη διάταξη, όμως, βελτιώνεται μόνο ελαφρά.

```
998 REM DIMINISHING INCREMENT SORT
999 REM (VERSION 2)
1000 N%=NUMBER%
1010 WHILE N%>1
1020 N%=(N%+2)/3
1030 FOR D%=N%+1 TO N%*2
1040 FOR E%=D% TO NUMBER% STEP N%
1050 FOR F%=E% TO D% STEP -N%
1060 IF A%(F%)<A%(F%-N%) THEN T%=A%(F%):
A%(F%)=A%(F%-N%):A%(F%-N%)=T% ELSE 1080
1070 NEXT
1080 NEXT E%
1090 NEXT
1100 WEND
```

```
1110 RETURN
```

Πρόγραμμα 5.5. Ταξινόμηση φθίνοντος βήματος (Παραλλαγή 2)

Αυτή η συγκεκριμένη παραλλαγή τροποποιήθηκε και χρησιμοποιήθηκε στο Πρόγραμμα 4.1 του προηγούμενου κεφαλαίου καί είναι απόλυτα αποδεκτή γραμμένη σε BASIC υπό την προϋπόθεση ότι το N δεν είναι πολύ μεγάλο. Παρακαλούμε σημειώστε ότι το σημείο διαίρεσης στη γραμμή 1020 είναι ακέραια διαίρεση, όχι κοινή διαίρεση.

Πίνακας 5.5. Διάρκειες ταξινόμησης φθίνοντος βήματος (Παραλλαγή 2).

Διάταξη πίνακα	Στοιχεία	Χρόνος εκτέλεσης
Αντίστροφη	100	9.92 δευτερόλεπτα
Αντίστροφη	200	19.82 δευτερόλεπτα
Αντίστροφη	300	37.14 δευτερόλεπτα
Σχεδόν διαταγμένη	100	7.36 δευτερόλεπτα
Σχεδόν διαταγμένη	200	14.66 δευτερόλεπτα
Σχεδόν διαταγμένη	300	26.11 δευτερόλεπτα
Τυχαία	100	11.78 δευτερόλεπτα
Τυχαία	200	23.85 δευτερόλεπτα
Τυχαία	300	45.13 δευτερόλεπτα

Η Quicksort (γρήγορη ταξινόμηση)

Ο αλγόριθμος της γρήγορης ταξινόμησης που επινοήθηκε και πήρε αυτό το όνομα από τον C. Hoare είναι ένας απ' τους πιο γρήγορους γνωστούς αλγόριθμους για να ταξινομήσουμε τυχαίους πίνακες στοιχείων και μπορεί να προσεγγίσει τον θεωρητικό ελάχιστο χρόνο ταξινόμησης ανάλογο προς $n\log_2 n$. Ο αλγόριθμος αυτός συχνά εντυπωσιάζει χάρις στην ταχύτητα στην ταξινόμηση μεγάλων αριθμών από τυχαία στοιχεία.

Η βασική ιδέα πίσω από την Ταχεία ταξινόμηση, είναι η τεχνική του «διαίρει και βασίλευε». Παρατηρήσαμε νωρίτερα ότι η ταξινόμηση μικρών λιστών είναι πολύ πιο αποτελεσματική από την ταξινόμηση μεγάλων. Έτσι, αν έχουμε έναν μεγάλο πίνακα και τον διασπάσουμε σε δύο υπολίστες που να περιέχουν διαφορετικές περιοχές αριθμών και ταξινομήσουμε την κάθε μία ξεχωριστά, θα γλυτώσουμε πολύ χρόνο ταξινόμησης. Για να το κάνουμε αυτό, εκτιμούμε ότι ένα στοιχείο του πίνακα θα έχει «κατ' ευτυχή συγκυρία» μία μέση τιμή και το ονομάζουμε *άξονα (pivot)*. Όλα τα στοιχεία του πίνακα που έχουν τιμή μικρότερη από τον άξονα θα τοποθετηθούν επάνω απ' αυτόν στο χαμηλότερο μισό του πίνακα και όλα τα στοιχεία που έ-

χουν τιμή μεγαλύτερη από τον άξονα θα τοποθετηθούν στο ανώτερο μισό. Αν αυτά τα δύο μέρη της διάταξης ταξινομηθούν ξεχωριστά, εκατέρωθεν του άξονα (pivot), τότε ο πίνακας θα ταξινομηθεί τελείως. 'Ολη η σημασία εδώ ευρίσκεται στην εκτίμηση της τιμής (του στοιχείου) που θα αποτελέσει τον άξονα. Για παράδειγμα, θα μπορούσαμε να διαλέξουμε το πρώτο στοιχείο του πίνακα ή το τελευταίο σ' ένα τυχαίο πίνακα. 'Ομως, αν η λίστα είναι μερικά ταξινομημένη, όπως μπορεί να συμβεί στην πράξη, το αποτέλεσμα μπορεί να είναι ιδιαίτερα απογοητευτικό γιατί ο άξονας θα απέχει πάρα πολύ από τη μέση τιμή ώστε να αξίζει η διάσπαση σε δύο υπολίστες. Αυτό το φαινόμενο μπορεί να μειωθεί στατιστικά εκλέγοντας τον άξονα σαν το στοιχείο που ευρίσκεται στη μέση ακριβώς του πίνακα. Υπάρχουν, βέβαια, πολλοί ακόμη τρόποι για να πάρουμε τον άξονα, αλλά εμείς θα υιοθετήσουμε αυτή τη μέθοδο. Παρεπιμπτόντως, κάτω και από τις χειρότερες ακόμα συνθήκες, όπου το παραπάνω φαινόμενο επικρατεί, ο χρόνος ταξινόμησης μπορεί να είναι τόσο άσχημος όσο για μία bubble sort (κατά προσέγγιση ανάλογος προς τη n^2) αλλά αυτό δεν είναι συνηθισμένο να συμβαίνει στην πράξη.

Μία μέθοδος για να υλοποιήσουμε την Quicksort είναι να διατηρήσουμε ξεχωριστές λίστες μ' αυτόν τον τρόπο μέχρι να έχουμε πολλές λίστες που να περιέχουν, ας πούμε, 15 στοιχεία το πολύ και μετά να τις ταξινομήσουμε με bubble sort. Εναλλακτικά, αν συνεχίσουμε και προχωρήσουμε αυτή την διαδικασία διαχωρισμού μέχρι το έσχατο όριο της, τότε κάθε υπολίστα θα περιλαμβάνει τελικά μόνο ένα στοιχείο, στην περίπτωση δε αυτή δεν θα υπάρχει ανάγκη να εφαρμόσουμε την bubble sort καθόλου.

Κανονικά, η Quicksort θα έπρεπε να γραφτεί σε μορφή κατάλληλη για να γίνεται εκμετάλλευση της *επαναληπτικότητας - παλινδρόμησης (recursion).* Ο όρος αυτός χρησιμοποιείται για να προσδιορίσει την πρακτική με την οποία ένα υποπρόγραμμα ανακαλεί τον εαυτό του μέσα από τα όρια του. Αυτός είναι ο τρόπος με τον οποίο πραγματοποιούνται τα πολλά επίπεδα διαχωρισμού. Η παλινδρόμηση κανονικά προϋποθέτει ότι διατίθενται καθορισμένες *ΔΙΑΔΙΚΑΣΙΕΣ (PROCEDURES), με τοπικές μεταβλητές (local variables).* Αυτές, αν και περιλαμβάνονται στις πλέον δομημένες γλώσσες μικροϋπολογιστών όπως η Pascal και η BBC BASIC, δυστυχώς λείπουν από την κατά τα άλλα εντυπωσιακή έκδοση της BASIC του Amstrad πιθανώς λόγω έλλειψης χώρου στην ROM. Αυτό πράγματι περιπλέκει τα πράγματα αλλά, ευτυχώς — αν και πέρα από τον σκοπό αυτού του βιβλίου — μπορεί να εφαρμοσθεί ένα σύνολο κανόνων για να επιτύχουμε την επαναληπτικότητα, εξάγοντας την από αλγορίθμους χρησιμοποιώντας έναν *σωρό (stack).* (Σωρός είναι ένας τομέας της μνήμης, όπου η τελευταία τιμή μεταβλητής που αποθηκεύεται είναι η πρώτη που ανακαλείται). Αυτοί οι κανόνες έχουν εφαρμοσθεί και αποτέλεσμα είναι το Πρόγραμμα 5.6. Τα όρια των δεικτών του πίνακα είναι p% και q% για την χαμηλότερη (επί κεφαλής και ουραγός αντίστοιχα) και r% και s% για την ανώτερη. Οι παράμετροι του επί κεφαλής και του ουραγού των υπολιστών που πρόκειται ακόμη να ταξινομηθούν τοποθετούνται στο σωρό. 'Ετσι αν προς το παρόν ταξινομούμε τον πίνακα μεταξύ A% (p%) και A% (q%), τότε τα r% και s% θα τοποθετηθούν στον σωρό, έτσι ώστε η υπολίστα A% (r%) έως A% (s%) θα μπορούσε να ταξινομηθεί αργότερα. 'Ετσι φαίνε-

ται ότι είναι καλύτερο να τοποθετήσουμε τα μακρύτερα όρια υπολίστας στον σωρό και να προχωρήσουμε στην διαδικασία για την συντομότερη υπολίστα αμέσως. Αυτός είναι ο σκοπός που εξυπηρετεί η γραμμή 1100. Κάθε φορά που μία λίστα διαχωρίζεται περαιτέρω στον εξωτερικό βρόχο WHILE / W-END η διαδικασία επαναλαμβάνεται. Στην έξοδο του βρόχου στη γραμμή 1120, οι υπολίστες, των οποίων οι παράμετροι τοποθετήθηκαν στον σωρό, λαμβάνονται διαδοχικά (τελευταίες εντός, πρώτες έξω) και ταξινομούνται με παρόμοιο τρόπο.

Υποστηρίζεται συχνά ότι η Quicksort χρησιμοποιεί πολλή μνήμη διότι η λίστα του προγράμματος είναι μακρύτερη, χρησιμοποιεί περισσότερες μεταβλητές και χρειάζεται να χρησιμοποιεί μία καθορισμένη περιοχή σωρού της μνήμης. Όμως, η επιπλέον μνήμη που χρησιμοποιείται από τον ίδιο τον σωρό δεν είναι υπερβολική. Ο αριθμός των επιπέδων σωρού που απαιτείται από την Quicksort δίνεται από τον $\log_2 n$. Έτσι για να ταξινομήσουμε, ας πούμε, 4096 αριθμούς, ο αριθμός των επιπέδων του σωρού που απαιτείται θα είναι $\log_2 (4096)=12$. Στο Πρόγραμμα 5.6 χρειαζόμαστε δύο σωρούς, που ν' αποτελούνται από τουλάχιστον 12 ακέραιους - στοιχεία της διάταξης ο καθένας, κάνοντας ένα σύνολο από 24. Κάθε στοιχείο του πίνακα που περιλαμβάνεται σε εντολή DIMENSION καταναλώνει 2 bytes έτσι θα απαιτηθούν 48 bytes χώρου του πίνακα. Αυτή η μνήμη δεν χρειάζεται να είναι μόνιμα διατιθέμενη γι' αυτόν τον σκοπό στον Amstrad αφού μπορούμε να εισάγουμε την εντολή DIMENSION για τον σωρό στην αρχή της υπορουτίνας (γραμμή 1000) και την εντολή ERASE στην γραμμή 1130 ελευθερώνοντας έτσι τη μνήμη ξανά για χρήση από το κύριο πρόγραμμα.

Το πρόγραμμα για την επίδειξη της Quicksort είναι το Πρόγραμμα 5.6 και ο πίνακας χρόνων ταξινόμησης είναι ο Πίνακας 5.6. Ο πίνακας αποκαλύπτει ότι, για το εύρος που εξετάσθηκε, υπάρχει μία σχεδόν γραμμική σχέση μεταξύ του χρόνου εκτέλεσης και του αριθμού των στοιχείων που ταξινομούνται. Η απόδοση για την σχεδόν διαταγμένη σειρά στοιχείων δεν ήταν ιδιαίτερα απογοητευτική στη συγκεκριμένη δοκιμή, αλλά μπορεί να μην είμαστε τόσο τυχεροί σε κάποιες από τις άλλες, που συναντιώνται πολύ συχνά σε περιπτώσεις κατάταξης που αναφέρθηκαν παραπάνω.

```
999 REM QUICKSORT SUBROUTINE
1000 DIM stack1%(16),stack2%(16)
1010 sp%=0:head%=1:tail%=NUMBER%
1020 WHILE head%<tail%
1030 pivot%=A%((head%+tail%)/2)
1040 a%=head%:b%=tail%
1050 WHILE A%(a%)<pivot%:a%=a%+1:WEND
1060 WHILE A%(b%)>pivot%:b%=b%-1:WEND
1070 IF a%<b% THEN t%=A%(a%):A%(a%)=A%(b
%):A%(b%)=t%:a%=a%+1:b%=b%-1:GOTO 1050
1080 IF a%=b% THEN q%=b%-1:r%=a%+1 ELSE
q%=b%:r%=a%
1090 sp%=sp%+1:p%=head%:s%=tail%
```

```
1100 IF q%-p%<s%-r% THEN stack1%(sp%)=r%
:stack2%(sp%)=s%:head%=p%:tail%=q% ELSE
stack1%(sp%)=p%:stack2%(sp%)=q%:head%=r%
:tail%=s%
1110 WEND
1120 IF sp%>0 THEN head%=stack1%(sp%):ta
il%=stack2%(sp%):sp%=sp%-1:GOTO 1020
1130 ERASE stack1%,stack2%
1140 RETURN
```

Πρόγραμμα 5.6. Quicksort.

Πίνακας 5.6. Διάρκειες Ταχείας ταξινόμησης (Quicksort).

Διάταξη πίνακα	Στοιχεία	Χρόνος εκτέλεσης
Αντίστροφη	100	6.08 δευτερόλεπτα
Αντίστροφη	200	12.77 δευτερόλεπτα
Αντίστροφη	300	18.65 δευτερόλεπτα
Σχεδόν καταταγμένη	100	7.45 δευτερόλεπτα
Σχεδόν καταταγμένη	200	15.51 δευτερόλεπτα
Σχεδόν καταταγμένη	300	23.30 δευτερόλεπτα
Τυχαία	100	10.12 δευτερόλεπτα
Τυχαία	200	19.32 δευτερόλεπτα
Τυχαία	300	30.41 δευτερόλεπτα
Τυχαία	1000	110.00 δευτερόλεπτα
Τυχαία	2000	243.48 δευτερόλεπτα

Όλοι συμφωνούν γενικά ότι απ' όλους τους αλγόριθμους ταξινόμησης δεν υπάρχει κάποιος που να υπερτερεί. Ο καθένας έχει τα σχετικά του πλεονεκτήματα και μειονεκτήματα. Ο προγραμματιστής θα πρέπει να διαλέξει εκείνον που ταιριάζει περισσότερο στον συγκεκριμένο προκείμενο σκοπό, λογαριάζοντας παράγοντες όπως ο αριθμός των στοιχείων για ταξινόμηση, διαθέσιμη μνήμη, αποδεκτός χρόνος εκτέλεσης, και —το πιο σημαντικό απ' όλα— ο κόπος για τον προγραμματισμό. Είναι άστοχο να χρησιμοποιήσουμε μία τελειοποιημένη ρουτίνα ταξινόμησης, όπως την Quicksort για να διατάξουμε μία λίστα ας πούμε είκοσι στοιχείων. Είναι εξ ίσου άστοχο να απασχολήσουμε μία τέτοια ρουτίνα για να ταξινομήσουμε μία αρκετά μεγάλη λίστα αλλά μόνο μια φορά. Μπορεί να βρεθεί μία αναλογία μεταξύ των ρουτινών ταξινόμησης και των καλαθοσφαιριστών. Οι καλαθοσφαιριστές είναι καλοί είτε κάτω απ' τα καλάθια είτε έξω από την «ρακέττα» αλλά σπάνια και τα δύο.

Ταξινόμηση αλφαριθμητικών

Όλα τα προηγούμενα παραδείγματα αυτού του κεφαλαίου χρησιμοποίησαν ακέραια στοιχεία πινάκων σαν μεταβλητές. Μπορούμε να αντικαταστήσουμε αυτά με αλφαριθμητικές μεταβλητές αλλάζοντας όλες τις εκφράσεις A% (N%) σε A$ (N%). Το αντίστοιχο τμήμα δοκιμών των προγραμμάτων, θα χρειαστεί βέβαια επίσης να αλλαχθεί. Σαν παράδειγμα της αναγκαίας μετατροπής, δημιουργήσαμε μία αλφαριθμητική έκδοση της Quicksort με ένα συνοδευτικό πρόγραμμα δοκιμών. Η λίστα δίδεται σαν Πρόγραμμα 5.7. Θα πρέπει να είναι μια ιδιαίτερα εύκολη διαδικασία να μετατρέψετε τις προηγούμενες υπορουτίνες για να ταξινομήσετε αλφαριθμητικά. Ένας πίνακας χρονικών διαρκειών, ο Πίνακας 5.7, δείχνει τους τυπικούς χρόνους εκτέλεσης για ταξινόμηση τυχαίων αλφαριθμητικών διατάξεων διαφόρων μεγεθών.

Πίνακας 5.7. Τυπικοί χρόνοι εκτέλεσης για το Πρόγραμμα 5.7.

Μέγεθος του πίνακα	Τυπικός χρόνος ταξινόμησης
100	9.65 δευτερόλεπτα
200	20:37 δευτερόλεπτα
300	31.41 δευτερόλεπτα
500	54.94 δευτερόλεπτα
1000	230.76 δευτερόλεπτα

```
10 REM SORT TEST PROGRAM
20 REM STRING ARRAY
30 CLS
40 INPUT"Sort how many strings";NUMBER%
50 PRINT
60 REM FILL AND DISPLAY RANDOM ARRAY
70 DIM A$(NUMBER%)
80 FOR R%=1 TO NUMBER%
90 B$=""
100 A%=6*RND+1
110 FOR Z%=1 TO A%
120 N%=25*RND
130 K$=CHR$(N%+65)
140 B$=B$+K$
150 NEXT
160 A$(R%)=B$
170 PRINT A$(R%)
180 NEXT
```

```
190 PRINT:PRINT
200 PRINT"SORTING ARRAY"
210 PRINT:PRINT
220 START=TIME/300
230 GOSUB 1000
240 T=TIME/300-START
250 FOR R%=1 TO NUMBER%
260 PRINT A$(R%)
270 NEXT
280 PRINT
290 PRINT"RECORDS SORTED=";NUMBER%
300 PRINT
310 PRINT"SORTING TIME=";ROUND(T,2);"SEC
ONDS"
320 END
330 '
340 '
999 REM STRING QUICKSORT SUBROUTINE
1000 DIM stack1%(16),stack2%(16)
1010 sp%=0:head%=1:tail%=NUMBER%
1020 WHILE head%<tail%
1030 pivot$=A$((head%+tail%)\2)
1040 a%=head%:b%=tail%
1050 WHILE A$(a%)<pivot$:a%=a%+1:WEND
1060 WHILE A$(b%)>pivot$:b%=b%-1:WEND
1070 IF a%<b% THEN t$=A$(a%):A$(a%)=A$(b
%):A$(b%)=t$:a%=a%+1:b%=b%-1:GOTO 1050
1080 IF a%=b% THEN q%=b%-1:r%=a%+1 ELSE
q%=b%:r%=a%
1090 sp%=sp%+1:p%=head%:s%=tail%
1100 IF q%-p%<s%-r% THEN stack1%(sp%)=r%
:stack2%(sp%)=s%:head%=p%:tail%=q% ELSE
stack1%(sp%)=p%:stack2%(sp%)=q%:head%=r%
:tail%=s%
1110 WEND
1120 IF sp%>0 THEN head%=stack1%(sp%):ta
il%=stack2(sp%):sp%=sp%-1:GOTO 1020
1130 ERASE stack1%,stack2%
1140 RETURN
```

Ορθογωνικός ή δισδιάστατος αλφαριθμητικός πίνακας

Ένας συνηθισμένος τρόπος οργάνωσης αρχείων, που συναντιέται σε

συστήματα αρχειοθέτησης βασισμένα στη RAM, είναι ο δισδιάστατος ή ορθογωνικός πίνακας. Αν σ' ένα αρχείο βασισμένο στη RAM δοθούν διαστάσεις (DIM) A$ (FIELDS%, RECORDS%), τότε το αρχείο A$ μπορεί να θεωρηθεί ότι περιέχει εγγραφές ίσες με RECORDS% που η καθεμιά τους έχει πεδία ίσα με FIELDS%. Οποιοδήποτε πεδίο μιας συγκεκριμένης εγγραφής μπορεί να καθοριστεί με την παράσταση A$ (F%, R%) όπου F% ο αριθμός του πεδίου και R% ο αριθμός της εγγραφής.

Αρχείο αποθηκευμένο στη RAM σαν ορθογωνικός πίνακας			
	Πεδίοϕ	Πεδίο 1	Πεδίο 2
Επικεφαλίδες πεδίων	A$(ϕ,ϕ)	A$(1,ϕ)	A$(2,ϕ)
Εγγραφή 1	A$(ϕ,1)	A$(1,1)	A$(2,1)
Εγγραφή 2	A$(ϕ,2)	A$(1,2)	A$(2,2)
Εγγραφή 3	A$(ϕ,3)	A$(1,3)	A$(2,3)
Εγγραφή 4	A$(ϕ,4)	A$(1,4)	A$(2,4)
Εγγραφή 5	A$(ϕ,5)	A$(1,5)	A$(2,5)
Εγγραφή n	A$(ϕ,n)	A$(1,n)	A$(2,n)

Σχ. 5.1. Αποθήκευση ενός αρχείου δεδομένων με τη μορφή ορθογωνικού πίνακα.

Το Σχήμα 5.1 δείχνει πώς εμφανίζεται στη μνήμη ένα αρχείο του τύπου αυτού. Υπάρχουν μερικά σημεία στο Σχ. 5.1 που χρειάζονται διευκρίνιση:

(α) Όταν η BASIC δίνει διαστάσεις σε έναν ορθογωνικό πίνακα, για κάθε στοιχείο του πίνακα κρατάει έναν *αλφαριθμητικό περιγραφέα* των 3 bytes ανεξάρτητα αν αυτά χρησιμοποιούνται ή όχι. Δυστυχώς, αυτό περιλαμβάνει και όλα τα στοιχεία που έχουν τιμή μηδέν. Για να μεγιστοποιήσουμε τη διαθέσιμη μνήμη μας, πρέπει να εξασφαλίσουμε ότι ανάμεσα στα αποδεκτά γι' αυτό το σκοπό πεδία θα είναι και το πεδίο μηδέν. Αν το κάνουμε αυτό, τότε

μπορούμε να δώσουμε διαστάσεις σε ένα, ας πούμε, αρχείο με 400 εγγραφές και 5 πεδία στην καθεμιά σαν A$ (4, 400) και όχι σαν A$(5, 400) όπως θα φαινόταν από πρώτη ματιά λογικότερο. Αν αμελήσουμε το σημείο αυτό, η συνέπεια θα είναι να σπαταλήσουμε 400 X 3 = 1200 byte μνήμης χωρίς να υπάρχει ανάγκη.

(β) Τα στοιχεία της εγγραφής μηδέν δεν σπαταλιούνται μια και προσφέρονται για ν' αποθηκεύσουμε εκεί τις ίδιες τις επικεφαλίδες των πεδίων.

Ταξινόμηση των ορθογωνικών αλφαριθμητικών πινάκων

Αν απαιτείται οι εγγραφές να ταξινομηθούν ανάλογα με ένα συγκεκριμένο πεδίο, πρέπει να τροποποιήσουμε τις ρουτίνες μας έτσι ώστε να εναλλάσσουν κάθε φορά όλα τα πεδία στα ζεύγη των εγγραφών. Για να γίνει αυτό θα χρειαστεί ένας επιπλέον βρόχος FOR/NEXT.

Για να εφαρμοστεί πετυχημένα μία αλφαριθμητική ταξινόμηση πάνω σε πεδία αριθμητικών δεδομένων, ο χρήστης πρέπει να εξασφαλίσει ότι όλες οι καταχωρήσεις του θα έχουν σταθερό αριθμό ψηφίων (μαζί με τα μηδενικά της αρχής). Μια εναλλακτική λύση σ' αυτό είναι η χρήση της εντολής VAL σαν συνθήκης. Ωστόσο, αυτό συνεπάγεται παραπέρα περιπλοκές όπως και τρομακτική αύξηση του χρόνου εκτέλεσης.

Το Πρόγραμμα 5.8 δείχνει πώς τροποποιείται η ταξινόμηση φθίνοντος βήματος (η εκδοχή 2) για να ταξινομήσει έναν ορθογωνικό πίνακα. Χρησιμοποιήσαμε αυτή την υπορουτίνα για το Πρόγραμμα 4.1 στο προηγούμενο κεφάλαιο. Δίνεται επίσης ένα πρόγραμμα ελέγχου που παράγει τυχαία έναν καθοριζόμενο αριθμό σταθερών, τυχαίων εγγραφών με τρία πεδία η καθεμιά. Στον Πίνακα 5.8 φαίνονται τυπικοί χρόνοι εκτέλεσης.

Πίνακας 5.8 Τυπικοί χρόνοι εκτέλεσης του Προγράμματος 5.8.

Εγγραφές	Τυπικός χρόνος ταξινόμησης
100	27.91 δευτερόλεπτα
200	109.66 δευτερόλεπτα
300	310.70 δευτερόλεπτα

```
10 REM TEST PROGRAM
20 REM SORTING A RECTANGULAR STRING
30 REM ARRAY WITH BASIC SUBROUTINES
40 CLS
50 INPUT"Sort how many 3 field records";
RECORDS%
60 FIELDS%=2:REM fields 0,1,2
70 INPUT"Sort which field (0-2)";FIELDNU
M%
80 PRINT
90 REM FILL AND DISPLAY RANDOM ARRAY
```

```
100 DIM A$(FIELDS%,RECORDS%)
110 FOR R%=1 TO RECORDS%
120 FOR F%=0 TO FIELDS%
130 B$=""
140 A%=6*RND+1
150 FOR Z%=1 TO A%
160 N%=25*RND
170 K$=CHR$(N%+65)
180 B$=B$+K$
190 NEXT
200 A$(F%,R%)=B$
210 PRINT A$(F%,R%),
220 NEXT
230 NEXT
240 PRINT:PRINT
250 PRINT"SORTING ARRAY"
260 PRINT:PRINT
270 START=TIME/300
280 GOSUB 1000
290 T=TIME/300-START
300 FOR R%=1 TO RECORDS%
310 FOR F%=1 TO FIELDS%
320 PRINT A$(F%,R%),
330 NEXT
340 NEXT
350 PRINT
360 PRINT"RECORDS SORTED=";RECORDS%
370 PRINT
380 PRINT"SORTING TIME=";ROUND(T,2);"SEC
ONDS"
390 END
400 '
410 '
998 REM DIMINISHING INCREMENT SORT
999 REM (VERSION 2)
1000 N%=RECORDS%
1010 WHILE N%>1
1020 N%=(N%+2)/3
1030 FOR D%=N%+1 TO N%*2
1040 FOR E%=D% TO RECORDS% STEP N%
1050 FOR R%=E% TO D% STEP -N%
1060 IF A$(FIELDNUM%,R%)<A$(FIELDNUM%,R%
-N%) THEN FOR C%=0 TO FIELDS%:T$=A$(C%,R
%):A$(C%,R%)=A$(C%,R%-N%):A$(C%,R%-N%)=T
$:NEXT ELSE 1080
```

```
1070 NEXT
1080 NEXT E%
1090 NEXT
1100 WEND
1110 RETURN
```

Πρόγραμμα 5.8 Ταξινόμηση φθίνοντος βήματος σ' έναν ορθογωνικό αλφαριθμητικό πίνακα.

Η παραλλαγή της Quicksott για ορθογωνικούς πίνακες, που δίνεται στο Πρόγραμμα 5.9, μπορεί να αντικαταστήσει τη ρουτίνα ταξινόμησης που χρησιμοποιήθηκε στο Πρόγραμμα 4.1 του προηγούμενου κεφάλαιου. Η συνολική επίδοση της παραλλαγής αυτής είναι καλύτερη, να θυμάστε όμως ότι πρέπει ν' αλλάξετε τις μεταβλητές RECORDS% και FIELDNUM% σε L% και F% αντίστοιχα. Ο Πίνακας 5.9 δείχνει τους τυπικούς χρόνους εκτέλεσης που πρέπει ν' αναμένονται.

Πίνακας 5.9 Τυπικοί χρόνοι εκτέλεσης για το Πρόγραμμα 5.9

Εγγραφές	Τυπικός χρόνος ταξινόμησης
100	15.49 δευτερόλεπτα
200	33.42 δευτερόλεπτα
300	144.87 δευτερόλεπτα
500	596.37 δευτερόλεπτα

```
998 REM SUBROUTINE:QUICKSORT
999 REM RECTANGULAR STRING ARRAY
1000 DIM stack1%(16),stack2%(16)
1010 sp%=0:head%=1:tail%=RECORDS%
1020 WHILE head%<tail%
1030 pivot$=A$(FIELDNUM%,(head%+tail%)/2
)
1040 a%=head%:b%=tail%
1050 WHILE A$(FIELDNUM%,a%)<pivot$:a%=a%
+1:WEND
1060 WHILE A$(FIELDNUM%,b%)>pivot$:b%=b%
-1:WEND
1070 IF a%<b% THEN FOR C%=0 TO FIELDS%:t
$=A$(C%,a%):A$(C%,a%)=A$(C%,b%):A$(C%,b%
)=t$:NEXT:a%=a%+1:b%=b%-1:GOTO 1050
1080 IF a%=b% THEN q%=b%-1:r%=a%+1 ELSE
q%=b%:r%=a%
1090 sp%=sp%+1:p%=head%:s%=tail%
1100 IF q%-p%<s%-r% THEN stack1%(sp%)=r%
:stack2%(sp%)=s%:head%=p%:tail%=q% ELSE
```

```
stack1%(sp%)=p%:stack2%(sp%)=q%:head%=r%
:tail%=s%
1100 IF q%-p%<s%-r% THEN stack1%(sp%)=r%
:stack2%(sp%)=s%:head%=p%:tail%=q% ELSE
stack1%(sp%)=p%:stack2%(sp%)=q%:head%=r%
:tail%=s%
1110 WEND
1120 IF sp%>0 THEN head%=stack1%(sp%):ta
il%=stack2%(sp%):sp%=sp%-1:GOTO 1020
1130 ERASE stack1%,stack2%
1140 RETURN
```

Πρόγραμμα 5.9. Η Quicksort σε έναν ορθογωνικό αλφαριθμητικό πίνακα.

Λύσεις με κώδικα μηχανής

Οποιαδήποτε τεχνική ταξινόμησης κι αν χρησιμοποιούμε, δεν μπορεί να αρνηθεί κανείς ότι όταν έχουμε να κάνουμε με μεγάλους αριθμούς στοιχείων, ο χρόνος εκτέλεσης μπορεί να γίνει απαράδεκτα μεγάλος. Στο κάτω - κάτω, βρισκόμαστε μπροστά σ' ένα απ' τα έμφυτα μειονεκτήματα της γλώσσας BASIC. Είναι γλώσσα *ερμηνευτική (interpretive)* και όχι *μεταφρασμένη (compiled)* και έτσι η ταχύτητα εκτέλεσης είναι μικρή. Η λύση στο πρόβλημα της ταχύτητας είναι ο κώδικας μηχανής.

Ακόμα και σε κώδικα μηχανής, εξακολουθεί να είναι πολύ σημαντικό το να διαλέξουμε έναν κατάλληλο αλγόριθμο. Για παράδειγμα, αν η λίστα μας είναι μεγάλη, μία bubblesort γραμμένη σε κώδικα μηχανής δεν θα εκτελεσθεί πολύ γρηγορότερα από μία Shellsort γραμμένη σε BASIC. Όμως, για να ελαχιστοποιήσουμε τη δυσκολία που έχει ο προγραμματισμός σε κώδικα μηχανής, συμφέρει μερικές φορές να διαλέγουμε αλγορίθμους που έχουν χαρακτήρα έμφυτα δυαδικό. Απ' όλους τους αλγόριμθους που σε συντομία εκθέσαμε στο κεφάλαιο αυτό, η ταξινόμηση φθίνοντος βήματος (εκδοχή 1) είναι ίσως η καλύτερη επιλογή, αν πάρουμε υπόψη μας τόσο τον κόπο στον προγραμματισμό όσο και τη γενική επίδοσή της. Αυτό είναι συνέπεια της δυαδικής μεθόδου που χρησιμοποιείται για να υπολογιστούν οι σειρές των βημάτων.

Παρ' όλο που το θέμα του προγραμματισμού σε κώδικα μηχανής Z80 ξεφεύγει απ' τα όρια του βιβλίου αυτού, θα ήταν κρίμα να μην συμπεριλαμβάναμε μερικές πολύ αποδοτικές ρουτίνες για ταξινόμηση πινάκων. Σας ζητάμε, επομένως, να αποδεχτείτε τις ρουτίνες αυτές με μία πολύ γενική μόνο εξήγηση. Επειδή όμως αξίζει τον κόπο να χρησιμοποιήσετε τις ρουτίνες αυτές, η έμφαση θα δοθεί στη χρήση τους και όχι στις λεπτομέρειες του προγραμματισμού τους. Αν όχι τίποτ' άλλο, θα πρέπει να πάρετε μία ιδέα των πλεονεκτημάτων που παρουσιάζει ως προς την ταχύτητα ο προγραμματισμός σε κώδικα μηχανής. (Σ.τ.μ. Όποιον ενδιαφέρει η εμβάθυνση στη

γλώσσα μηχανής υπάρχει το βιβλίο «Γλώσσα Μηχανής για αρχαρίους στον Amstrad», εκδ. Κλειδάριθμος).

'Ολες οι λίστες σε *γλώσσα assembly* του τμήματος αυτού έχουν αναπτυχθεί χρησιμοποιώντας τον «Hisoft DEVPAC Editor, Assembler, Disassembler & Monitor» (Amsoft 116). Μην ανησυχείτε αν δεν μπορείτε να βρείτε αυτό το συγκεκριμένο πρόγραμμα. Σας δίνουμε και προγράμματα για φόρτωμα σε δεκαεξαδικό γραμμένα σε BASIC, που θα επιτρέψουν να φορτωθούν τα bytes του κώδικα μηχανής απ' ευθείας από εντολές DATA της BASIC, σε μία καθορισμένη περιοχή της μνήμης. Αυτό το τμήμα της μνήμης μπορεί κατόπιν να σωθεί σε κασέτα (ή δισκέτα) σαν δυαδικό αρχείο.

Ταξινόμηση αλφαριθμητικού πίνακα με κώδικα μηχανής

Τα πρακτικά αρχεία έχουν συνήθως περισσότερα πεδία που περιέχουν αλφαριθμητικούς χαρακτήρες απ' ότι αριθμητικούς. 'Οταν ο interpreter της BASIC δίνει διαστάσεις σε έναν αλφαριθμητικό πίνακα, φτιάχνεται ένας πίνακας προσπελάσεως που περιέχει 3 bytes για κάθε αλφαριθμητικό στοιχείο. Αυτά τα bytes δεν είναι οι ίδιες οι αλφαριθμητικές, αλλά το μήκος και η διεύθυνση που είναι αποθηκευμένη η αλφαριθμητική. Αυτά τα τρία bytes ονομάζονται *περιγραφείς της αλφαριθμητικής (string descriptors)* και αντιπροσωπεύουν το μήκος της αλφαριθμητικής, τη διεύθυνση του χαμηλότερου byte και τη διεύθυνση του υψηλότερου byte αντίστοιχα. Το μήκος της αλφαριθμητικής είναι το χαμηλότερο στη μνήμη. Αυτή η ίδια η αλφαριθμητική, αποτελούμενη από τους κωδικούς ASCII σε διαδοχικές θέσεις της μνήμης, αποθηκεύεται στην διεύθυνση εκκίνησης που δίνεται στα byte 2 και 3 που πιο πάνω αναφέρθηκαν. 'Ετσι, ένας αλφαριθμητικός πίνακας σχηματίζεται από μία σειρά τέτοιων περιγραφέων πού είναι αποθηκευμένοι σειριακά στη μνήμη. Η εναλλαγή των αλφαριθμητικών κατά τη διάρκεια μιας ταξινόμησης δεν είναι τόσο δύσκολη όσο ίσως φαίνεται με μία πρώτη ματιά. Μπορούμε ν' αφήσουμε τις ίδιες τις αλφαριθμητικές να είναι εκεί που βρίσκονται στη μνήμη και να εναλλάξουμε τους περιγραφείς τους, μια και αυτοί λένε στον interpreter της BASIC πού είναι αποθηκευμένες οι αλφαριθμητικές.

Το Πρόγραμμα 5.10 είναι αυτό που είναι γνωστό σαν λίστα σε κώδικα assembly ή *πηγαίο κώδικα (source code)* μιας ταξινόμησης φθίνοντος βήματος αλφαριθμητικής. Ο τελικός κώδικας μηχανής ή *αντικειμενικός κώδικας (object code)* που είναι απαραίτητος για την ταξινόμηση αλφαριθμητικών πινάκων, παράγεται από αυτόν και τοποθετείται στη μνήμη από τη διεύθυνση &9980 και πέρα.

Το αντίστοιχο πρόγραμμα ελέγχου είναι το Πρόγραμμα 5.11 το οποίο δημιουργεί έναν τυχαίο αλφαριθμητικό πίνακα, καλεί τη ρουτίνα κώδικα μηχανής και εμφανίζει τον ταξινομημένο πίνακα. Τέλος, για εκείνους που δεν έχουν assembler δίνεται στο Πρόγραμμα 5.12 ένα πρόγραμμα για φόρτωμα σε δεκαεξαδικό.

```
10 ;DIMINISHING INCREMENT SORT
```

```
20 ;OF A STRING ARRAY
30 begin:     EQU   #9980
40 top:       EQU   begin+#F0
50 number:    EQU   top
60 array:     EQU   top+2
70 power:     EQU   top+4
80 cycles:    EQU   top+5
90 count:     EQU   top+7
100 flag:     EQU   top+9
110 point:    EQU   top+10
120 point2:   EQU   top+12
130 len1:     EQU   top+14
140 len2:     EQU   top+15
150 addr1:    EQU   top+16
160 addr2:    EQU   top+18
170 size:     EQU   top+20
180           ORG   begin
190 ;Initialise
200           LD    HL,1
210           LD    (size),HL
220 ;Pick up AND store
230 ;array start address
240           LD    L,(IX)
250           LD    H,(IX+1)
260           LD    (array),HL
270 ;Pick up number of strings
280 ;AND store in number
290           LD    L,(IX+2)
300           LD    H,(IX+3)
310           LD    (number),HL
320 :Find NEXT power of 2 >= number
330 ;AND store in size
340           EX    DE,HL
350           SUB   A
360 loop:     INC   A
370           LD    HL,(size)
380           ADD   HL,HL
390           LD    (size),HL
400           SBC   HL,DE
410           JR    C,loop
420 ;Store corresponding
430 ;index in power
440           LD    (power),A
450 ;Divide size by 2
460 outer:    LD    DE,(size)
```

```
470             SRL    D
480             RR     E
490             LD     (size),de
500 ;Subtract size from number
510 ;AND store in cycles
520             LD     HL,(number)
530             SBC    HL,DE
540             LD     (cycles),HL
550 ;Initialise count AND swop flag
560 mid:        SUB    A
570             LD     (count),A
580             LD     (count+1),A
590             LD     (flag),A
600 ;Store array start address
610 ;in point1
620             LD     HL,(array)
630             LD     (point1),HL
640 ;Multiply size by 3 add point1
650 ;store result in point2
660             LD     HL,(size)
670             LD     D,H
680             LD     E,L
690             ADD    HL,HL
700             ADD    HL,DE
710             LD     DE,(point1)
720             ADD    HL,DE
730             LD     (point2),HL
740 ;Get lengths AND addresses
750 ;of strings TO be compared
760 inner:      LD     HL,(point1)
770             LD     A,(HL)
780             LD     (len1),A
790             INC    HL
800             LD     A,(HL)
810             LD     (addr1),A
820             INC    HL
830             LD     A,(HL)
840             LD     (addr1+1),A
850             LD     HL,(point2)
860             LD     A,(HL)
870             LD     (len2),A
880             INC    HL
890             LD     A,(HL)
900             LD     (addr2),A
910             INC    HL
```

```
920              LD     A,(HL)
930              LD     (addr2+1),A
940              LD     B,0
950 ;Compare strings a character at
960 ;a TIME,branch swop OR noswop
970              LD     DE,(addr2)
980              LD     HL,(addr1)
990 comp:        LD     A,(DE)
1000             CP     (HL)
1010             JR     C,swop
1020             JR     NZ,noswop
1030             INC    B
1040             LD     A,(len1)
1050             CP     B
1060             JR     Z,noswop
1070             LD     A,(len2)
1080             CP     B
1090             JR     Z,swop
1100             INC    DE
1110             INC    HL
1120             JR     NZ,comp
1130 ;Initialise byte counter
1140 ;set swop flag
1150 swop:       LD     B,3
1160             LD     (flag),A
1170             LD     DE,(point1)
1180             LD     HL,(point2)
1190 ;Swop string descriptors
1200 ;a byte at a TIME
1210 loop2:      LD     A,(DE)
1220             LD     C,(HL)
1230             EX     DE,HL
1240             LD     (HL),C
1250             LD     (DE),A
1260             INC    DE
1270             INC    HL
1280             DJNZ   loop2
1290 ;Add 3 TO each of
1300 ;point1 AND point2
1310 noswop:     LD     DE,3
1320             LD     HL,(point1)
1330             ADD    HL,DE
1340             LD     (point1),HL
1350             LD     HL,(point2)
1360             ADD    HL,DE
```

```
1370          LD    (point2),HL
1380 ;Increment count
1390          LD    DE,(count)
1400          INC   DE
1410          LD    (count),DE
1420 ;Compare count TO cycles
1430 ;IF NOT equal jump TO inner
1440          LD    HL,(cycles)
1450          SBC   HL,DE
1460          JP    NZ,inner
1470 ;Jump TO fclear
1480 ;IF swop flag is CLEAR
1490          LD    A,(flag)
1500          CP    0
1510          JR    Z,fclear
1520 ;Decrement cycles AND
1530 ;jump TO mid IF NOT zero
1540          LD    HL,(cycles)
1550          DEC   HL
1560          LD    (cycles),HL
1570          LD    A,H
1580          OR    L
1590          JP    NZ,mid
1600 ;Decrement power AND
1610 ;jump TO outer IF NOT zero
1620 fclear: LD    HL,power
1630          DEC   (HL)
1640          JP    NZ,outer
1650          RET
```

Πρόγραμμα 5.10 Ταξινόμηση φθίνοντος βήματος σε αλφαριθμητικό πίνακα.

```
10 REM SORT TEST PROGRAM
20 REM STRING ARRAY
30 CLS
40 PRINT"Place object code tape in DATAC
ORDER"
50 OPENOUT"buffer"
60 MEMORY &997F
70 CLOSEOUT
80 LOAD"STRINGSORT",&9980
90 CLS
100 INPUT"Sort how many strings";NUMBER%
110 PRINT
```

```
120 REM FILL AND DISPLAY RANDOM ARRAY
130 DIM A$(NUMBER%)
140 FOR R%=0 TO NUMBER%-1
150 B$=""
160 A%=6*RND+1
170 FOR Z%=1 TO A%
180 N%=25*RND
190 K$=CHR$(N%+65)
200 B$=B$+K$
210 NEXT
220 A$(R%)=B$
230 PRINT A$(R%)
240 NEXT
250 PRINT:PRINT
260 PRINT"SORTING ARRAY"
270 PRINT:PRINT
280 START=TIME/300
290 CALL &9980,NUMBER%,@A$(0)
300 T=TIME/300-START
310 FOR R%=0 TO NUMBER%-1
320 PRINT A$(R%)
330 NEXT
340 PRINT
350 PRINT"RECORDS SORTED=":NUMBER%
360 PRINT
370 PRINT"SORTING TIME=":ROUND(T,2);"SEC
ONDS"
380 PRINT
390 INPUT"Another test (Y/N)";K$
400 K$=UPPER$(K$)
410 IF K$="Y" THEN ERASE A$:GOTO 90 ELSE
 END
```

Πρόγραμμα 5.11 Πρόγραμμα ελέγχου σε BASIC για ταξινόμηση αλφαριθμητικού πίνακα με κώδικα μηχανής.

```
10 REM PRODUCING A MACHINE CODE FILE
20 SORT STRING ARRAY
30 REM (no assembler needed)
40 CLS
50 ADDRESS%=&9980
60 FOR N%=0 TO &EF
70 READ BYTE$
80 POKE ADDRESS%+N%,VAL("&"+BYTE$)
90 NEXT
```

```
100 PRINT"Now produce an object code fil
e on tape"
110 SAVE"STRINGSORT",B,&9980,&F0
120 END
130 DATA 21,01,00,22,84,9A,DD,6E
140 DATA 00,DD,66,01,22,72,9A,DD
150 DATA 6E,02,DD,66,03,22,70,9A
160 DATA EB,97,3C,2A,84,9A,29,22
170 DATA 84,9A,ED,52,38,F4,32,74
180 DATA 9A,ED,5B,84,9A,CB,3A,CB
190 DATA 1B,ED,53,84,9A,2A,70,9A
200 DATA ED,52,22,75,9A,97,32,77
210 DATA 9A,32,78,9A,32,79,9A,2A
220 DATA 72,9A,22,7A,9A,2A,84,9A
230 DATA 54,5D,29,19,ED,5B,7A,9A
240 DATA 19,22,7C,9A,2A,7A,9A,7E
250 DATA 32,7E,9A,23,7E,32,80,9A
260 DATA 23,7E,32,81,9A,2A,7C,9A
270 DATA 7E,32,7F,9A,23,7E,32,82
280 DATA 9A,23,7E,32,83,9A,06,00
290 DATA ED,5B,82,9A,2A,80,9A,1A
300 DATA BE,38,13,20,26,04,3A,7E
310 DATA 9A,B8,28,1F,3A,7F,9A,B8
320 DATA 28,04,13,23,20,E9,06,03
330 DATA 32,79,9A,ED,5B,7A,9A,2A
340 DATA 7C,9A,1A,4E,EB,71,12,13
350 DATA 23,10,F7,11,03,00,2A,7A
360 DATA 9A,19,22,7A,9A,2A,7C,9A
370 DATA 19,22,7C,9A,ED,5B,77,9A
380 DATA 13,ED,53,77,9A,2A,75,9A
390 DATA ED,52,C2,DC,99,3A,79,9A
400 DATA FE,00,28,0C,2A,75,9A,2B
410 DATA 22,75,9A,7C,B5,C2,BD,99
420 DATA 21,74,9A,35,C2,A9,99,C9
```

Πρόγραμμα 5.12 Πρόγραμμα για φόρτωμα σε δεκαεξαδικό για αλφαριθμητική ταξινόμηση σε κώδικα μηχανής.

Παραγωγή ενός αρχείου αντικειμενικού κώδικα

Για να χρησιμοποιήσουμε άνετα υπορουτίνες κώδικα μηχανής από πρόγραμμα BASIC είναι αναγκαίο να φτιάξουμε ένα *δυαδικό αρχείο* του αντικειμενικού κώδικα σε κασέτα ή δισκέτα. Αυτό το αρχείο μπορεί μετά να φορτωθεί αυτόματα από το πρόγραμμα BASIC στις λίγες πρώτες γραμμές. Υπάρ-

χουν δύο τρόποι για την παραγωγή του αντικειμενικού κώδικα.

Μέθοδος 1: Χρησιμοποιώντας έναν assembler

Η μέθοδος αυτή προϋποθέτει ότι έχετε στη διάθεσή σας τον Hisoft DEVPAC Assembler.

(α) Φορτώστε τον Hisoft DEVPAC Assembler στην, ας πούμε, διεύθυνση 6500 - δεκαδική. Η διεύθυνση θα σας ζητηθεί με σήμα αναμονής.

(β) Πληκτρολογήστε τη λίστα του κώδικα assembly ακριβώς όπως είναι τυπωμένη στο Πρόγραμμα 5.10 (αυτό ορισμένες φορές λέγεται λίστα του πηγαίου κώδικα).

(γ) Σώστε ένα αντίγραφο του πηγαίου κώδικα σε κασέτα, πληκτρολογώντας:

P 10, 1650, <όνομα αρχείου>

Το αρχείο μπορεί αργότερα να ξαναφορτωθεί με:

G,, <όνομα αρχείου>

(δ) Μετατρέψτε τον πηγαίο κώδικα σε assembly πληκτρολογώντας A σαν διαταγή του assembler. Όταν το πρόγραμμα σας ζητήσει το μέγεθος του πίνακα, απαντήστε 500. Ο χώρος αυτός είναι παραπάνω από αρκετός για τον πίνακα των συμβόλων.

(ε) Αν σας αναφερθούν τυχόν λάθη πληκτρολόγησης, διορθώστε τα.

(στ) Φτιάξτε ένα αρχείο αντικειμενικού κώδικα που να μπορεί να φορτωθεί σε ένα κύριο πρόγραμμα κατευθείαν από κασέτα (τα ίδια ισχύουν και για δισκέτα). Αυτό γίνεται αυτόματα πληκτρολογώντας την παρακάτω διαταγή του assembler:

O,,STRINGSORT

Προσέξτε ότι το όνομα αρχείου STRINGSORT δεν είναι κλεισμένο μέσα σε εισαγωγικά.

(ζ) Κλείστε και ξανανοίξτε τον υπολογιστή ώστε να καθαρίσει η μνήμη.

(η) Πληκτρολογήστε το πρόγραμμα ελέγχου, το Πρόγραμμα 5.11. Αυτό αυτομάτως θα φορτώσει, θα τρέξει και θα ελέγξει το αρχείο STRINGSORT που μόλις έχετε φτιάξει.

Ο ίδιος ο αντικειμενικός κώδικας φορτώνεται σε μία περιοχή της μνήμης μεγέθους 512 byte που είναι διαφυλαγμένη πάνω από την HIMEM στο &9980 αλλά κάτω από την μονίμως διατεθειμένη περιοχή για την προσωρινή μνήμη της κασέτας (Ξανακοιτάξτε το «Διάθεση περιοχής στην προσωρινή μνήμη της κασέτας» στο Κεφάλαιο 1).

Μέθοδος 2: Χωρίς assembler

Πληκτρολογήστε το Πρόγραμμα 5.12 και τρέξτε το. Το αρχειο του αντικειμενικού κώδικα θα παραχθεί αυτόματα. Το πρόγραμμα αυτό είναι απλά ένας βρόχος που παίρνει τα bytes του κώδικα μηχανής από τις προτάσεις DATA και τα μεταφέρει κατ' ευθείαν στην μνήμη απ' το &9980 και παραπέρα. Μετά ο κώδικας σώζεται αυτόματα σε κασέτα (δισκέτα).

Επανατοποθέτηση του κώδικα μηχανής

Το βασικό μειονέκτημα της μεθόδου 2 είναι ότι ο κώδικας δεν θα εκτελεστεί παρά στη διεύθυνση &9980. Αν τον μετακινήσουμε κάπου αλλού στη μνήμη το αποτέλεσμα θα είναι η δημιουργία... χάους, γιατί ο αντικειμενικός κώδικας είναι όπως λέμε μη επανατοποθετήσιμος. Αν έχετε ένα κανονικό μηχάνημα χωρίς εξωτερικούς οδηγούς δισκέτας, φωτεινές πένες κ.λπ. δεν υπάρχει πρόβλημα. Ωστόσο, ίσως φανεί αναγκαίο να καταφύγετε στη Μέθοδο 1 για να ξαναμετατρέψετε τον πηγαίο κώδικα σε assembly σε μία εναλλακτική διεύθυνση αν συνδεθούν τέτοιες περιφερειακές μονάδες. Με το Πρόγραμμα 5.10 αυτό δεν είναι δύσκολο: απλώς αλλάξτε τη διεύθυνση στην πρόταση EQU της γραμμής 30 και ξανακάντε τη μετατροπή.

Κλήση της STRINGSORT *από οποιοδήποτε πρόγραμμα*

Για να καλέσετε τη ρουτίνα STRINGSORT που είναι σε κώδικα μηχανής είναι αναγκαίο να περάσετε μερικές παραμέτρους. Αυτό επιτυγχάνεται αν στο πρόγραμμά σας παρεμβάλετε μία γραμμή. Για παράδειγμα, το

```
CALL &9980, NUMBER%, @A$(1)
```

θα ταξινομήσει τον πίνακα A$ απ' το A$(1) ως το A$(NUMBER%).

Το &9980 είναι η διεύθυνση εκτέλεσης της υπορουτίνας. Αυτό μπορείτε να το αλλάξετε, αν η μετατροπή του κώδικα σε assembly γίνει σε άλλη θέση της μνήμης. Η NUMBER%, μία ακέραια μεταβλητή, είναι ο αριθμός των αλφαριθμητικών του πίνακα που πρόκειται να ταξινομηθούν. Οποιαδήποτε άλλη μεταβλητή δουλεύει εδώ, αρκεί να έχει την ίδια τιμή. Για παράδειγμα, θα μπορούσατε αντί γι' αυτήν να χρησιμοποιήσετε την SIZE%. Η παράμετρος A$(1) περνάει τη διεύθυνση του πρώτου αλφαριθμητικού περιγραφέα του πίνακα που πρόκειται να ταξινομηθεί.

Αν περάσετε την παράμετρο @A$(0) θα ταξινομηθούν τα στοιχεία από A$(0) ως και A$(NUMBER% - 1). Αν περάσετε την @A$(1) θα ταξινομηθούν τα στοιχεία από A$(1) ως και A$(NUMBER%). Να σημειωθεί ότι το σύμβολο «@» πριν από μία μεταβλητή σημαίνει ότι θα περαστεί στη ρουτίνα του κώδικα μηχανής η διεύθυνση της μεταβλητής, όχι η ίδια η τιμή της μεταβλητής.

Όταν εκτελείται σε BASIC μία εντολή CALL, οι τιμές της λίστας των παραμέτρων που ακολουθούν τη διεύθυνση εκτέλεσης, περνιώνται στην υπορουτίνα του κώδικα μηχανής. Αυτές αποθηκεύονται αυτόματα σε ένα

μπλοκ παραμέτρων που έχει αντιγραφεί από μία διεύθυνση αποθηκευμένη στον καταχωρητή IX του μικροεπεξεργαστή Z80. Οι τιμές της λίστας των παραμέτρων είναι αντίγραφα των περιεχομένων του καταχωρητή IX σε σειρά αντίστροφη εκείνης που εμφανίζεται στην εντολή CALL. Για παράδειγμα, στην παραπάνω CALL, η διεύθυνση του αλφαριθμητικού περιγραφέα A$(1) καταχωρείται στις θέσεις (IX) και (IX+1) και η τιμή της NUMBER% στις (IX+2) και (IX+3) - θέσεις του byte κατώτερης τάξης και του byte ανώτερης τάξης αντίστοιχα. Αυτές οι παράμετροι μπορεί μετά να παρθούν από κει και να αποθηκευτούν σ' ένα πιο βολικό τμήμα της μνήμης, από την ίδια την υπορουτίνα του κώδικα μηχανής.

Ο Πίνακας 5.10 σας δίνει μίαν ιδέα των χρόνων εκτέλεσης που πρέπει να αναμένονται απ' το Πρόγραμμα 5.10.

Πίνακας 5.10 Τυπικοί χρόνοι εκτέλεσης του Προγράμματος 5.10

Μέγεθος πίνακα	Τυπικοί χρόνοι ταξινόμησης
100	0.53 δευτερόλεπτα
200	1.59 δευτερόλεπτα
300	2.97 δευτερόλεπτα
500	7.23 δευτερόλεπτα
1000	22.32 δευτερόλεπτα
2000	64.35 δευτερόλεπτα

Ταξινόμηση ορθογωνικού αλφαριθμητικού πίνακα με τη μέθοδο του φθίνοντος βήματος σε κώδικα μηχανής.

Το Πρόγραμμα 5.13 θα ταξινομήσει έναν δισδιάστατο αλφαριθμητικό πίνακα του τύπου που περιγράψαμε νωρίτερα στο κεφάλαιο αυτό. Είναι πολύ γρήγορο. Στην πραγματικότητα, μπορεί να ταξινομήσει έναν πολύ μεγάλο αριθμό εγγραφών σε 8 περίπου δευτερόλεπτα. Ωστόσο, για να λειτουργήσει πετυχημένα η μέθοδος, πρέπει στην εντολή καθορισμού διαστάσεων ο δείκτης του πεδίου να δίνεται πρώτος. Δηλαδή, αν ο ορθογωνικός πίνακας έχει πάρει διαστάσεις DIM A$(FIELDS%, RECORDS%) και τα μεμονωμένα πεδία F% ενώ οι εγγραφές R%, πρέπει να γράφουμε A%(F%, R%).

```
10 ;DIMINISHING INCREMENT SORT OF A
20 ;TWO DIMENSIONAL STRING ARRAY
30 begin:     EQU   #9980
40 top:       EQU   begin+#11 C
50 number:    EQU   top
60 array:     EQU   top+2
70 power:     EQU   top+4
80 cycles:    EQU   top+5
90 bytes:     EQU   top+7
```

```
100 count:    EQU   top+9
110 flag:     EQU   top+11
120 point1:   EQU   top+12
130 point2:   EQU   top+14
140 len1:     EQU   top+16
150 len2:     EQU   top+17
160 addr1:    EQU   top+18
170 addr2:    EQU   top+20
180 size:     EQU   top+22
190 offset:   EQU   top+24
200           ORG   begin
210 ;Initialise
220           LD    HL,0
230           LD    (bytes),HL
240           LD    (offset),HL
250           INC   HL
260           LD    (size),HL
270 ;Pick up AND store
280 ;array start adress
290           LD    L,(IX)
300           LD    H,(IX+1)
310           LD    (array),HL
320 ;Pick up field number
330 ;multiply by 3 AND
340 ;stor in offset
350           LD    A,(IX+2)
360           LD    B,A
370           SLA   A
380           ADD   A,B
390           LD    (offset),A
400 ;Pick up number of fields
410 ;add 1 AND store in bytes
420           LD    A,(IX+4)
430           INC   A
440           LD    B,A
450           SLA   A
460           ADD   A,B
470           LD    (bytes),A
480 ;Pick up number of records
490 ;AND store in number
500           LD    L,(IX+6)
510           LD    H,(IX+7)
520           LD    (number),HL
530 ;Find NEXT power of 2 >= number
540 ;AND store in size
```

```
550             EX      DE,HL
560             SUB     A
570 loop:       INC     A
580             LD      HL,(size)
590             ADD     HL,HL
600             LD      (size),HL
610             SBC     HL,DE
620             JR      C,loop
630 ;store corresponding
640 ;index in power
650             LD      (power),A
660 ;Divide size by 2
670 outer:      LD      DE,(size)
680             SRL     D
690             RR      E
700             LD      (size),DE
710 ;Subtract size from number
720 ;AND store in cycles
730             LD      HL,(number)
740             SBC     HL,DE
750             LD      (cycles),HL
760 ;Initialise count AND swop flag
770 mid:        SUB     A
780             LD      (count),A
790             LD      (count+1),A
800             LD      (flag),A
810 ;Store array start address
820 ;in point1
830             LD      HL,(array)
840             LD      (point1),HL
850 ;Multiply size by bytes add
860 ;point1 store result in point2
870             LD      A,(bytes)
880             LD      B,A
890             LD      HL,0
900             LD      DE,(size)
910 mult:       ADD     HL,DE
920             DJNZ    mult
930             LD      DE,(point1)
940             ADD     HL,DE
950             LD      (point2),HL
960 ;Get lengths AND addresses of
970 ;strings TO be compared
980 inner:      LD      DE,(offset)
990             LD      HL,(point1)
```

```
1000            ADD   HL,DE
1010            LD    A,(HL)
1020            LD    (len1),A
1030            INC   HL
1040            LD    A,(HL)
1050            LD    (addr1),A
1060            INC   HL
1070            LD    A,(HL)
1080            LD    (addr1+1),A
1090            LD    HL,(point2)
1100            ADD   HL,DE
1110            LD    A,(HL)
1120            LD    (len2),A
1130            INC   HL
1140            LD    A,(HL)
1150            LD    (addr2),A
1160            INC   HL
1170            LD    A,(HL)
1180            LD    (addr2+1),A
1190            LD    B,0
1200            LD    DE,(addr2)
1210            LD    HL,(addr1)
1220 ;Compare strings a character at
1230 ;a TIME branch swop OR noswop
1240 comp:      LD    A,(DE)
1250            CP    (HL)
1260            JR    C,swop
1270            JR    NZ,noswop
1280            INC   B
1290            LD    A,(len1)
1300            CP    B
1310            JR    Z,noswop
1320            LD    A,(len2)
1330            CP    B
1340            JR    Z,swop
1350            INC   DE
1360            INC   HL
1370            JR    NZ,comp
1380 ;Initialise byte counter
1390 ;set swop flag
1400 swop:      LD    A,(bytes)
1410            LD    B,A
1420            LD    (flag),A
1430 ;Swop string descriptors FOR
1440 ;whole dimension a byte at a TIME
```

```
1450            LD      DE,(point1)
1460            LD      HL,(poin2)
1470 loop2:     LD      A,(DE)
1480            LD      C,(HL)
1490            EX      DE,HL
1500            LD      (HL),C
1510            LD      (DE),A
1520            INC     DE
1530            INC     HL
1540            DJNZ    loop2
1550 ;Add bytes TO each of
1560 ;point1 AND point2
1570 noswop:    LD      DE,(bytes)
1580            LD      HL,(point1)
1590            ADD     HL,DE
1600            LD      (point1),HL
1610            LD      HL,(point2)
1620            ADD     HL,DE
1630            LD      (point2),HL
1640 ;Increment count
1650            LD      DE,(count)
1660            INC     DE
1670            LD      (count),DE
1680 ;Compare count TO cycles
1690 ;IF NOT equal jump TO inner
1700            LD      HL,(cycles)
1710            SBC     HL,DE
1720            JP      NZ,inner
1730 ;Jump TO fclar
1740 ;IF swop flag is CLEAR
1750            LD      A,(flag)
1760            CP      0
1770            JR      Z,fclear
1780 ;Decrement cycles AND
1790 ;jump TO mid IF NOT zero
1800            LD      HL,(cycles)
1810            DEC     HL
1820            LD      (cycles),HL
1830            LD      A,H
1840            OR      L
1850            JP      NZ,mid
1860 ;Decrement power AND
1870 ;jump TO outer IF NOT zero
1880 fclear:    LD      HL,power
1890            DEC     (HL)
```

```
1900            JP    NZ,outer
1910            RET
```

Πρόγραμμα 5.13 Ταξινόμηση ορθογωνικού αλφαριθμητικού πίνακα με κώδικα μηχανής.

Το Πρόγραμμα 5.14 είναι ένα πρόγραμμα ελέγχου που καταρτίζει ένα τυχαίο αρχείο με εγγραφές των 3 πεδίων για να ελέγξει τη λειτουργία του Προγράμματος 5.13. Τα 3 πεδία αριθμούνται πεδίο μηδέν, πεδίο 1 και πεδίο 2 αντίστοιχα και ο πίνακας παίρνει διαστάσεις DIM (FIELDS%, RECORDS%), όπου το FIELDS% έχει πάρει τιμή 2. Αν αυτό σας φαίνεται παράξενο, δείτε τις παρατηρήσεις που κάναμε νωρίτερα σ' αυτό το κεφάλαιο σε σχέση με τη χρήση των ορθογωνικών πινάκων. Σας ζητείται να δώσετε τον αριθμό των εγγραφών και ποιό πεδίο (από 0 ως 2) πρόκειται να γίνει αντικείμενο της ταξινόμησης. Όπως και πριν, η ρουτίνα μπορεί να κληθεί όχι μόνο απ' το πρόγραμμα ελέγχου της BASIC αλλά και από οποιοδήποτε άλλο πρόγραμμα χρησιμοποιώντας:

```
CALL &9980, RECORDS%, FIELDS%, FIELDNUM%,
@A$(0,1)
```

Οι παράμετροι που απαιτούνται είναι:

(1) Η διεύθυνση εκτέλεσης. Αυτή είναι &9980 αν ο πηγαίος κώδικας μετατρέπεται σε assembly έτσι όπως καθορίζει η EQU της γραμμής 30 του Προγράμματος 5.13.

(2) Η RECORDS% μπορεί να είναι είτε ο μέγιστος αριθμός των εγγραφών σύμφωνα με τις διαστάσεις του αρχείου (υποθέτοντας ότι το αρχείο είναι γεμάτο) ή ο αριθμός των εγγραφών που αυτή τη στιγμή βρίσκονται στον πίνακα.

(3) Η FIELDS% είναι ο αριθμός των πεδίων μειωμένος κατά ένα ή η τιμή στην οποία έχουν πάρει διαστάσεις τα πεδία.

(4) Η FIELDNUM% είναι ο αριθμός του πεδίου που ταξινομείται (0 ως η-1) όπου η ο συνολικός αριθμός των πεδίων). Η ρουτίνα μπορεί να ταξινομήσει εγγραφές που έχουν μέχρι και 85 πεδία, ένα όριο που ξεπερνά κατά πολύ τις απαιτήσεις των περισσότερων αρχείων!

(5) Η @A$(0,1) είναι η διεύθυνση του στοιχείου του πίνακα που χρησιμοποιείται πρώτο, δηλαδή εκείνου του πεδίου της πρώτης εγγραφής που χρησιμοποιείται πρώτο. Τα μηδενικά στοιχεία της διάστασης της εγγραφής χρησιμοποιούνται σαν επικεφαλής πεδίων και έτσι δεν θα πρέπει να περιληφθούν στην ταξινόμηση.

```
10 REM TEST PROGRAM
20 REM MACHINE CODE SORT OF A
30 REM RECTANGULAR STRING ARRAY
40 CLS
```

```
50 PRINT"Place object code tape in DATAC
ORDER"
60 OPENOUT"buffer"
70 MEMORY &997F
80 CLOSEOUT
90 LOAD"SORT",&9980
100 CLS
110 INPUT"Sort how many 3 field records"
;RECORDS%
120 FIELD%=2:REM 3 fields (0,1 & 2)
130 INPUT"Sort which field (0-2)";FIELDN
UM%
140 IF FIELDNUM%<0 OR FIELDNUM%>2 THEN 1
30
150 PRINT
160 REM FILL AND DISPLAY RANDOM ARRAY
170 DIM A$(FIELDS%,RECORDS%)
180 FOR R%=1 TO RECORDS%
190 FOR F%=0 TO FIELDS%
200 B$=""
210 A%=6*RND+1
220 FOR Z%=1 TO A%
230 N%=25*RND
240 K$=CHR$(N%+65)
250 B$=B$+K$
260 NEXT
270 A$(F%,R%)=B$
280 PRINT A$(F%,R%),
290 NEXT
300 NEXT
310 PRINT:PRINT
320 PRINT"SORTING ARRAY"
330 PRINT:PRINT
340 START=TIME/300
350 CALL &9980,RECORDS%,FIELDS%,FIELDNUM
%,@A$(0,1)
360 T=TIME/300-START
370 FOR R%=1 TO RECORDS%
380 FOR F%=0 TO FIELDS%
390 PRINT A$(F%,R%),
400 NEXT
410 NEXT
420 PRINT
430 PRINT"RECORDS SORTED=";RECORDS%
440 PRINT
```

```
450 PRINT"SORTING TIME=";ROUND(T,2);"SEC
ONDS"
460 PRINT
470 INPUT"Another test (Y/N)";K$
480 K$=UPPER$(K$)
490 IF K$="Y" THEN ERASE A$:GOTO 100 ELS
E END
```

Πρόγραμμα 5.14. Πρόγραμμα ελέγχου για την ταξινόμηση ορθογωνικού πίνακα με κώδικα μηχανής.

Παραγωγή ενός αρχείου αντικειμενικού κώδικα του Προγράμματος 5.13

Ο πηγαίος κώδικας μετατρέπεται αυτόματα σε αντικειμενικό κώδικα στη διεύθυνση που εμείς του καθορίζουμε. Στο πρόγραμμα της λίστας αυτή τυχαίνει να είναι η &9980 και παραπέρα, όπως καθορίστηκε από τη γραμμή 30. Τώρα χρειαζόμαστε ένα αντίγραφο αυτού του αντικειμενικού κώδικα σε κασέτα (ή δισκέτα) για να το χρησιμοποιούμε με οποιοδήποτε άλλο πρόγραμμα. Και πάλι υπάρχουν δύο μέθοδοι, ανάλογα με το αν υπάρχει assembler διαθέσιμος ή όχι. Παρακάτω δίνονται σύντομες οδηγίες για το πώς να το κάνετε αυτό. Για πιο γενικές οδηγίες και σχόλια ανατρέξτε στο παράδειγμα του STRINGSORT που δώσαμε παραπάνω.

Μέθοδος 1: Χρησιμοποιώντας τον Hisoft DEVPAC Assembler

(α) Φορτώστε τον Assembler π.χ. στη διεύθυνση &6500 δεκαδική.
(β) Πληκτρολογήστε το Πρόγραμμα 5.13.
(γ) Σώστε τον πηγαίο κώδικα πληκτρολογώντας:

P 10, 1910, < όνομα αρχείου >

(δ) Μετατρέψτε τον πηγαίο κώδικα σε assembly δίνοντας τη διαταγή A του assembler. Όταν σας ζητηθεί μέγεθος πίνακα, απαντήστε με 50

(ε) Σώστε τον αντικειμενικό κώδικα σε κασέτα (ή δισκέτα) πληκτρολογώντας την διαταγή:

0,,SORT

(στ) Ανοίξτε και κλείστε τον υπολογιστή και μετά πληκτρολογήστε το Πρόγραμμα 5.14. Αυτό το πρόγραμμα θα φορτώσει, θα τρέξει και θα ελέγξει το αρχείο αντικειμενικού κώδικα που μόλις έχετε φτιάξει.

Μέθοδος 2: Χωρίς assembler

Απλώς πληκτρολογήστε και τρέξτε το Πρόγραμμα 5.15. Αυτό το πρόγραμμα παράγει ένα αρχείο αντικειμενικού κώδικα που ονομάζεται «SORT» κατευθείαν. Δυστυχώς, ο κώδικας μηχανής που δημιουργείται μ' αυτό τον τρόπο δεν μπορεί να επανατοποθετηθεί και έτσι εκτελείται μόνο στη διεύθυνση &9980.

```
10 REM PRODUCING A MACHINE CODE FILE
20 REM RECTANGULAR STRING ARRAY SORT
30 REM (no assembler needed)
40 CLS
50 ADDRESS%=&9980
60 FOR N%=0 TO &11B
70 READ BYTE$
80 POKE ADDRESS%+N%,VAL("&"+BYTE$)
90 NEXT
100 PRINT"Now produce an object code fil
e on tape"
110 SAVE"SORT",B,&9980,&11C
120 END
130 DATA 21,00,00,22,A4,9A,22,B5
140 DATA 9A,23,22,B3,9A,DD,6E,00
150 DATA DD,66,01,22,9F,9A,DD,7E
160 DATA 02,47,CB,27,80,32,B5,9A
170 DATA DD,7E,04,3C,47,CB,27,80
180 DATA 32,A4,9A,DD,6E,06,DD,66
190 DATA 07,22,9D,9A,EB,97,3C,2A
200 DATA B3,9A,29,22,B3,9A,ED,52
210 DATA 38,F4,32,A1,9A,ED,5B,B3
220 DATA 9A,CB,3A,CB,1B,ED,53,B3
230 DATA 9A,2A,9D,9A,ED,52,22,A2
240 DATA 9A,97,32,A6,9A,32,A7,9A
250 DATA 32,A8,9A,2A,9F,9A,22,A9
260 DATA 9A,3A,A4,9A,47,21,00,00
270 DATA ED,5B,B3,9A,19,10,FD,ED
280 DATA 5B,A9,9A,19,22,AB,9A,ED
290 DATA 5B,B5,9A,2A,A9,9A,19,7E
300 DATA 32,AD,9A,23,7E,32,AF,9A
310 DATA 23,7E,32,B0,9A,2A,AB,9A
320 DATA 19,7E,32,AE,9A,23,7E,32
330 DATA B1,9A,23,7E,32,B2,9A,06
340 DATA 00,ED,5B,B1,9A,2A,AF,9A
350 DATA 1A,BE,38,13,20,28,04,3A
360 DATA AD,9A,B8,28,21,3A,AE,9A
370 DATA B8,28,04,13,23,20,E9,3A
380 DATA A4,9A,47,32,A8,9A,ED,5B
```

```
390 DATA A9,9A,2A,AB,9A,1A,4E,EB
400 DATA 71,12,13,23,10,F7,ED,5B
410 DATA A4,9A,2A,A9,9A,19,22,A9
420 DATA 9A,2A,AB,9A,19,22,AB,9A
430 DATA ED,5B,A6,9A,13,ED,53,A6
440 DATA 9A,2A,A2,9A,ED,52,C2,FF
450 DATA 99,3A,A8,9A,FE,00,28,0C
460 DATA 2A,A2,9A,2B,22,A2,9A,7C
470 DATA B5,C2,D9,99,21,A1,9A,35
480 DATA C2,C5,99,C9
```

Πρόγραμμα 5.15 Πρόγραμμα για φόρτωμα σε δεκαεξαδικό για την ταξινόμηση του ορθογωνικού αλφαριθμητικού πίνακα σε κώδικα μηχανής.

Ο Πίνακας 5.11 σας βοηθάει να πάρετε μία ιδέα του χρόνου ταξινόμησης που πρέπει να αναμένεται για διάφορους αριθμούς τυχαίων εγγραφών (των 3 πεδίων).

Πίνακας 5.11. Τυπικοί χρόνοι εκτέλεσης του Προγράμματος 5.13

Μέγεθος πίνακα	Τυπικοί χρόνοι ταξινόμησης
100	0.53 δευτερόλεπτα
200	1.70 δευτερόλεπτα
300	3.21 δευτερόλεπτα
500	7.45 δευτερόλεπτα

Βελτίωση του συστήματος αρχειοθέτησης του Κεφαλαίου 4

Το ολοκληρωμένο πρόγραμμα συστήματος αρχειοθέτησης που περιγράψαμε πιο πριν στο Κεφάλαιο 4, έχει πολλά να επωφεληθεί από την αυξημένη ταχύτητα της υπορουτίνας ταξινόμησης σε κώδικα μηχανής. Οι λιγοστές τροποποιήσεις που απαιτούνται δίνονται παρακάτω:

(1) Αντικαταστήστε την υπάρχουσα υπορουτίνα SORT FILE του Προγράμματος 4.1 (απ' τη γραμμή 1480 ως και την 1620) με τις γραμμές που δίνονται παρακάτω:

```
1480 REM SORT FILE
1490 GOSUB 2480
1500 IF L%<2 THEN 1520
1510 CALL &9980,L%,fields%,F%,@A$(0,1)
1520 RETURN
```

Η γραμμή 1500 παρακάμπτει το CALL αν στο αρχείο υπάρχουν λιγότερες από 2 εγγραφές. Η γραμμή 1510 καλεί τη ρουτίνα του κώδικα μηχανής και περνάει τις αντίστοιχες μεταβλητές απ' το Πρόγραμμα 4.1.

(2) Αντικαταστήστε τις πρώτες 4 γραμμές του Προγράμματος 4.1 (γραμμές 10 ως και 40) με τις παρακάτω 7 γραμμές:

```
10 REM CASSETTE MULTIFILING SYSTEM
20 REM WITH MACHINE CODE SORT ROUTINE
30 OPENOUT"buffer"
40 MEMORY &997F
50 CLOSEOUT
60 CLS:PRINT"Now load machine code sort
routine"
70 LOAD"SORT",&9980
```

Οι γραμμές 20 ως 30 θέτουν τη HIMEM στην διεύθυνση &997F, διαφυλάσσοντας έτσι 512 bytes (ποσόν μάλλον γενναιόδωρο) για την υπορουτίνα του κώδικα μηχανής. Ταυτόχρονα, στις προσωρινές μνήμες της κασέτας (ή της δισκέτας) διατίθεται χώρος με μόνιμη ρύθμιση. Ο κώδικας μηχανής φορτώνεται στη γραμμή 40.

Βολεύει να τα κανονίσετε έτσι που αμέσως μετά το αντίγραφο του Προγράμματος 4.1 να είναι στην κασέτα τοποθετημένο ένα αντίγραφο του αρχείου αντικειμενικού κώδικα. Αυτό κάνει δυνατό να μπορεί το Πρόγραμμα 4.1 να φορτώνει τον κώδικα αμέσως χωρίς να χρειάζεται ο χρήστης να αλλάζει κασέτες στο Datacorder.

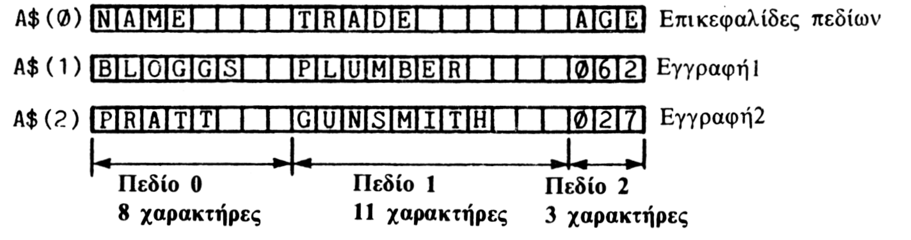
Χρήση αλφαριθμητικών Πινάκων σαν αρχεία δεδομένων.

Όταν χρησιμοποιούμε αλφαριθμητικούς πίνακες για χειρισμό δεδομένων είναι αναγκαίο να τονίσουμε ένα-δύο σημεία που αφορούν την ταξινόμηση.

Πρώτον, μια και όλα τα δεδομένα του αρχείου αποθηκεύονται σ' έναν και μόνο δισδιάστατο αλφαριθμητικό πίνακα, όλα τα αριθμητικά πεδία πρέπει να αποθηκευτούν και να ταξινομηθούν με τη μορφή της *αλφαριθμητικής τους αναπαράστασης*. Επομένως, αν πρόκειται να ταξινομήσουμε εγγραφές με αριθμητικά πεδία πρέπει να εξασφαλίσουμε, στη φάση της εισαγωγής των δεδομένων, ότι όλα τα αριθμητικά δεδομένα ενός δεδομένου πεδίου έχουν τον ίδιο αριθμό ψηφίων. Αυτό συχνά θα έχει τη συνέπεια να βάλουμε μερικά μηδενικά μπροστά απ' τον αριθμό. Κάτι τέτοιο συμβαίνει συχνά σε εφαρμογές της καθημερινής ζωής, π.χ. λογιστικά. Δεύτερον, αν έχετε πεδία που ν' αντιπροσωπεύουν χρηματικές τιμές ίσως είναι αναγκαίο να αποθηκεύσετε τα π.χ. $28 σαν $00028.00 για να μπορείτε μετά να τα ταξινομήσετε. Αυτό γίνεται γιατί άλλες καταχωρήσεις του ίδιου πεδίου μπορεί να περιέχουν μία τιμή λίγων σεντς.

Ταξινόμηση καθορισμένου μήκους εγγραφών πολλαπλών πεδίων με κώδικα μηχανής.

Υπάρχουν δύο βασικοί τρόποι για να δημιουργήσουμε εγγραφές πολλαπλών πεδίων που να καταλαμβάνουν μόνιμη θέση στη RAM. Ο πιο κοινός είναι αυτός που μόλις εξετάσαμε, ο δισδιάστατος πίνακας όπου οι εγγραφές καταλαμβάνουν τη μία διάσταση και τα πεδία την άλλη. Μία ολότελα διαφορετική μέθοδος είναι αυτή κατά την οποία ολόκληρο το πεδίο αποθηκεύεται σαν ένας αλφαριθμητικός πίνακας στον οποίο το κάθε πεδίο/εγγραφή έχει καθορισμένο μήκος. Το μήκος των αλφαριθμητικών των εγγραφών και των πεδίων καθορίζεται ίσο με την μέγιστη πιθανή τιμή. Ύστερα, τα τυχόν αχρησιμοποίητα τμήματα εγγραφών/πεδίων «γεμίζονται» με κενά. Για παράδειγμα, το Σχ. 5.2 δείχνει τις επικεφαλίδες των πεδίων και ένα ζευγάρι εγγραφών από ένα απλό αρχείο και πώς γεμίζονται με κενά. Το σημείο έναρξης κάθε πεδίου είναι σ' ένα σημείο της αλφαριθμητικής που είναι σταθερό για όλες τις εγγραφές.



Σχ. 5.2. Ένα ζευγάρι εγγραφές καθορισμένου μήκους με πολλαπλά πεδία.

Αυτή η τεχνική είναι μερικές φορές πιο οικονομική απ' την άποψη της χρησιμοποιούμενης μνήμης μια και για κάθε εγγραφή χρειάζεται να οριστεί μόνο ένας αλφαριθμητικός περιγραφέας, ενώ στον δισδιάστατο πίνακα απαιτείται ένας περιγραφέας για το κάθε πεδίο της εγγραφής. Τα βασικότερα μειονεκτήματα της μεθόδου, ωστόσο, είναι ο περιορισμός στους 255 χαρακτήρες ανά εγγραφή και η σπατάλη της RAM που χρησιμοποιείται για το «παραγέμισμα» των συντομότερων καταχωρήσεων με κενά.

```
10 ;DIMINISHING INCREMENT SORT OF
20 ;MULTIFIELD FIXED LENGTH RECORDS
30 begin:     EQU   #9980
40 top:       EQU   begin+#F5
50 number:    EQU   top
60 array:     EQU   top+2
70 power:     EQU   top+4
80 cycles:    EQU   top+5
90 count:     EQU   top+7
100 flag:     EQU   top+9
110 point1:   EQU   top+10
120 point2:   EQU   top+12
130 length:   EQU   top+14
140 start:    EQU   top+15
```

```
150 addr1:    EQU   top+16
160 addr2:    EQU   top+18
170 size:     EQU   top+20
180           ORG   begin
190 ;Initialise
200           LD    HL,1
210           LD    (size),HL
220 ;Pick up AND store
230 ;array start address
240           LD    L,(IX)
250           LD    H,(IX+1)
260           LD    (array),HL
270 ;Pick up AND store
280 ;field length in length
290           LD    A,(IX+2)
300           LD    (length),A
310 ;Pick up field start position
320 ;decrement AND store in start
330           LD    A,(IX+4)
340           DEC   A
350           LD    (start),A
360 ;Pick up number of records
370 ;AND store in number
380           LD    L,(IX+6)
390           LD    H,(IX+7)
400           LD    (number),HL
410 ;Find NEXT power of 2>= number
420 ;AND store in size
430           EX    DE,HL
440           SUB   A
450 loop:     INC   A
460           LD    HL,(size)
470           ADD   HL,HL
480           LD    (size),HL
490           SBC   HL,DE
500           JR    C,loop
510 ;Store corresponding
520 ;index in power
530           LD    (power),A
540 ;Divide size by 2
550 outer:    LD    DE,(size)
560           SRL   D
570           RR    E
580           LD    (size),DE
590 ;Subtract size from number
```

```
600 ;AND store in cycles
610             LD     HL,(number)
620             SBC    HL,DE
630             LD     (cycles),HL
640 ;Initialise count AND swop flag
650 mid:        SUB    A
660             LD     (count),A
670             LD     (count+1),A
680             LD     (flag),A
690 ;Store array start address
700 ;in point1
710             LD     HL,(array)
720             LD     (point1),HL
730 ;Multiply size by 3 add point1
740 ;store result in point2
750             LD     HL,(size)
760             LD     D,H
770             LD     E,L
780             ADD    HL,HL
790             ADD    HL,DE
800             LD     DE,(point1)
810             ADD    HL,DE
820             LD     (point2),HL
830 ;Get addresses of strings
840 ;TO be compared
850 inner:      LD     HL,(point1)
860             INC    HL
870             LD     A,(HL)
880             LD     (addr1),A
890             INC    HL
900             LD     A,(HL)
910             LD     (addr1+1),A
920             LD     HL,(point2)
930             INC    HL
940             LD     A,(HL)
950             LD     (addr2),A
960             INC    HL
970             LD     A,(HL)
980             LD     (addr2+1),A
990 ;Add start TO ech of string
1000 ;character start addresses
1010            LD     A,(start)
1020            LD     C,A
1030            LD     B,0
1040            LD     HL,(addr2)
```

```
1050          ADD   HL,BC
1060          EX    DE,HL
1070          LD    HL,(addr1)
1080          ADD   HL,BC
1090 ;Compare substrings a character
1100 ;at a TIME branch swop OR noswop
1110 ;finish when length is reached
1120 comp:    LD    A,(DE)
1130          CP    (HL)
1140          JR    C,swop
1150          JR    NZ,noswop
1160          INC   B
1170          LD    A,(length)
1180          CP    B
1190          JR    Z,noswop
1200          INC   DE
1210          INC   HL
1220          JR    NZ,comp
1230 ;Initialise byte counter
1240 ;set swop flag
1250 swop:    LD    B,3
1260          LD    (flag),A
1270          LD    DE,(point1)
1280          LD    HL,(point2)
1290 ;Swop string descriptors
1300 ;a byte at a TIME
1310 loop2:   LD    A,(DE)
1320          LD    C,(HL)
1330          EX    DE,HL
1340          LD    (HL),C
1350          LD    (DE),A
1360          INC   DE
1370          INC   HL
1380          DJNZ  loop2
1390 ;Add 3 TO each of
1400 ;point1 AND point2
1410 noswop:  LD    DE,3
1420          LD    HL,(point1)
1430          ADD   HL,DE
1440          LD    (point1),HL
1450          LD    HL,(point2)
1460          ADD   HL,DE
1470          LD    (point2),HL
1480 ;Increment count
1490          LD    DE,(count)
```

```
1500           INC   DE
1510           LD    (count),DE
1520 ;Compare count TO cycles
1530 ;IF NOT equal jump TO inner
1540           LD    HL,(cycles)
1550           SBC   HL,DE
1560           JP    NZ,inner
1570 ;Jump TO fclear
1580 ;IF swop flag is CLEAR
1590           LD    A,(flag)
1600           CP    0
1610           JR    Z,fclear
1620 ;Decrement cycles AND
1630 ;jump TO mid IF NOT zero
1640           LD    HL,(cycles)
1650           DEC   HL
1660           LD    (cycles),HL
1670           LD    A,H
1680           OR    L
1690           JP    NZ,mid
1700 ;Decrement power AND
1710 ;jump TO outer IF NOT zero
1720 fclear:   LD    HL,power
1730           DEC   (HL)
1740           JP    NZ,outer
1750           RET
```

Πρόγραμμα 5.16 Ταξινόμηση φθίνοντος βήματος σε καθορισμένου μήκους εγγραφές με πολλαπλά πεδία.

Η συνολική δομή είναι όμοια μ' αυτήν που είδαμε στις προηγούμενες ρουτίνες, ενώ οι επιπλέον κώδικες σχολιάζονται στο πρόγραμμα. Ο κώδικας μηχανής μπορεί να εκτελεστεί από οποιοδήποτε πρόγραμμα με την εντολή CALL

CALL &9980, NUMBER%, start%, fieldlength%, A$(1)

όπου:

ο όρος &9980 είναι η διεύθυνση εκτέλεσης της ρουτίνας,
NUMBER% = ο τρέχων αριθμός εγγραφών του πίνακα,
start% = η θέση που αρχίζουν οι χαρακτήρες του πεδίου κατά το οποίο πρόκειται να ταξινομηθούν οι εγγραφές. Επιτρεπτό εύρος από 1 ως 255,
fieldlength% = ο αριθμός των χαρακτήρων του πεδίου κατά το οποίο πρόκειται να ταξινομηθούν οι εγγραφές.

⊡A$(1) = το στοιχείο του πίνακα που χρησιμοποιείται πρώτο. Το μηδέν μπορεί να αφεθεί για επικεφαλίδες και να εξαιρεθεί απ' την ταξινόμηση.

Το Πρόγραμμα 5.17 είναι ένα πρόγραμμα ελέγχου που παράγει τυχαία αλφαριθμητικές σειρές των 20 χαρακτήρων. Ο αλφαριθμητικός πίνακας μπορεί να ταξινομηθεί κατά οποιαδήποτε καθορισμένου μήκους αλφαριθμητική υπο-σειρά που να αρχίζει σε οποιαδήποτε σταθερή θέση χαρακτήρα. Αυτό μιμείται την ταξινόμηση ενός αρχείου δομημένου σύμφωνα με τις αρχές του Σχ. 5.2. Για να αποκρυπτογραφείται πιο εύκολα η ταξινομημένη λίστα, το πρόγραμμα ελέγχου παρεμβάλλει κενά ανάμεσα στα όρια των πεδίων που καθορίσαμε.

```
10 REM SORT TEST PROGRAM
20 REM MULTIFIELD FIXED LENGTH RECORDS
30 CLS
40 PRINT"Place object code tape in DATAC
ORDER"
50 OPENOUT"buffer"
60 MEMORY &997F
70 CLOSEOUT
80 LOAD"MFSORT",&9980
90 CLS
100 length%=20
110 INPUT"Sort how many 20 character rec
ords";NUMBER%
120 INPUT"Enter field START position (1-
20)";start%
130 IF start%<1 OR start%>length% THEN 1
20
140 INPUT"Enter field LENGTH (1-20)";fie
ldlength%
150 IF fieldlength%<1 OR start%+fieldlen
gth%>length%+1 THEN 140
160 PRINT
170 REM FILL AND DISPLAY RANDOM ARRAY
180 DIM A$(NUMBER%)
190 FOR R%=0 TO NUMBER%-1
200 B$=""
210 FOR Z%=1 TO length%
220 N%=25*RND
230 K$=CHR$(N%+65)
240 B$=B$+K$
250 NEXT
260 A$(R%)=B$
270 PRINT A$(R%)
```

```
280 NEXT
290 PRINT:PRINT
300 PRINT"SORTING ARRAY"
310 PRINT:PRINT
320 START=TIME/300
330 CALL &9980,NUMBER%,start%,fieldlengt
h%,@A(0)
340 T=TIME/300-START
350 FOR R%=0 TO NUMBER%-1
360 PRINT LEFT$(A$(R%),start%-1);SPC(1);
MID$(A$(R%),start%,fieldlength%);SPC(1);
MID$(A$(R%),start%+fieldlength%,LEN(A$(R
%)))
370 NEXT
380 PRINT
390 PRINT"RECORDS SORTED=";NUMBER%
400 PRINT
410 PRINT"SORTING TIME=";ROUND(T,2);"SEC
ONDS"
420 PRINT
430 INPUT"Another test (Y/N)"K$
440 K$=UPPER$(K$)
450 IF K$="Y" THEN ERASE A$:GOTO 90 ELSE
 END
```

Πρόγραμμα 5.17. Πρόγραμμα ελέγχου για την ταξινόμηση αλφαριθμητικής πολλαπλών πεδίων σταθερού μήκους.

Παραγωγή αρχείου αντικειμενικού κώδικα του Προγράμματος 5.16

Όπως στα προηγούμενα παραδείγματα, υπάρχουν δύο τρόποι να φτιάξουμε το αρχείο αντικειμενικού κώδικα που χρειάζεται για χρήση από άλλο πρόγραμμα. Δίνουμε μία γενική περιγραφή παρακάτω, αλλά για περισσότερες λεπτομέρειες ξανακοιτάξτε την περιγραφή της απλής αλφαριθμητικής ταξινόμησης με κώδικα μηχανής.

Μέθοδος 1: Χρησιμοποιώντας τον Hisoft DEVPAC assembler

(α) Φορτώστε τον assembler στην διεύθυνση, έστω, 6500 δεκαδικό.

(β) Πληκτρολογήστε το Πρόγραμμα 5.16 όπως είναι στη λίστα που δίνουμε.

(γ) Σώστε τον πηγαίο κώδικα σε κασέτα (δισκέτα) πληκτρολογώντας:

P 10, 1750, <όνομα αρχείου>

(δ) Μετατρέψτε τον πηγαίο κώδικα σε assembly δίνοντας τη διαταγή A. Όταν σας ζητηθεί μέγεθος πίνακα, απαντήστε με 500.

(ε) Σώστε ένα αντίγραφο του αντικειμενικού κώδικα σε κασέτα (δισκέτα) πληκτρολογώντας τη διαταγή του assembler:

0,,MFSORT

(στ) Κλείστε και ανοίξτε τον υπολογιστή και μετά πληκτρολογήστε το Πρόγραμμα 5.17. Αυτό θα φορτώσει, θα τρέξει και θα ελέγξει το αρχείο αντικειμενικού κώδικα που έχει παραχθεί.

Μέθοδος 2: Χωρίς assembler

Για να φτιάξετε ένα αρχείο αντικειμενικού κώδικα που φορτώνει και εκτελείται σε διεύθυνση &9982 δεν έχετε παρά να πληκτρολογήσετε το Πρόγραμμα 5.18 και να το τρέξετε.

```
10 REM PRODUCING A MACHINE CODE FILE
20 REM SORT STRING ARRAY OF
30 REM FIXED LENGTH RECORDS
40 REM (so assembler needed)
50 CLS
60 ADDRESS%=&9980
70 FOR N%=0 TO &F4
80 READ BYTE$
90 POKE ADDRESS%+N%,VAL("&"+BYTE$)
100 NEXT
110 PRINT"Now produce an object code fil
e on tape"
120 SAVE"MFSORT",B,&9980,&F5
130 END
140 DATA 21,01,00,22,89,9A,DD,6E
150 DATA 00,DD,66,01,22,77,9A,DD
160 DATA 7E,02,32,83,9A,DD,7E,04
170 DATA 3D,32,84,9A,DD,6E,06,DD
180 DATA 66,07,22,75,9A,EB,97,3C
190 DATA 2A,89,9A,29,22,89,9A,ED
200 DATA 52,38,F4,32,79,9A,ED,5B
210 DATA 89,9A,CB,3A,CB,1B,ED,53
220 DATA 89,9A,2A,75,9A,ED,52,22
230 DATA 7A,9A,97,32,7C,9A,32,7D
240 DATA 9A,32,7E,9A,2A,77,9A,22
250 DATA 7F,9A,2A,89,9A,54,5D,29
260 DATA 19,ED,5B,7F,9A,19,22,81
```

```
270 DATA 9A,2A,7F,9A,23,7E,32,85
280 DATA 9A,23,7E,32,86,9A,2A,81
290 DATA 9A,23,7E,32,87,9A,23,7E
300 DATA 32,88,9A,3A,84,9A,4F,06
310 DATA 00,2A,87,9A,09,EB,2A,85
320 DATA 9A,09,1A,BE,38,0D,20,20
330 DATA 04,3A,83,9A,B8,28,19,13
340 DATA 23,20,EF,06,03,32,7E,9A
350 DATA ED,5B,7F,9A,2A,81,9A,1A
360 DATA 4E,EB,71,12,13,23,10,F7
370 DATA 11,03,00,2A,7F,9A,19,22
380 DATA 7F,9A,2A,81,9A,19,22,81
390 DATA 9A,ED,5B,7C,9A,13,ED,53
400 DATA 7C,9A,2A,7A,9A,ED,52,C2
410 DATA E9,99,3A,7E,9A,FE,00,28
420 DATA 0C,2A,7A,9A,2B,22,7A,9A
430 DATA 7C,B5,C2,CA,99,21,79,9A
440 DATA 35,C2,B6,99,C9
```

Πρόγραμμα 5.18 Πρόγραμμα για φόρτωμα σε δεκαεξαδικό του αντικειμενικού κώδικα του Προγράμματος 5.16

Αναζήτηση σε πίνακες

Η αναζήτηση μιας συγκεκριμένης εγγραφής μέσα σ' ένα αρχείο είναι μία συνηθισμένη επεξεργαστική απαίτηση, κοινότερη ακόμα κι απ' την ταξινόμηση. Ο σκοπός της αναζήτησης είναι να βρεθεί το δεδομένο μαζί με το στοιχείο-κλειδί που το προσδιορίζει. Παρ' όλο που στα παραδείγματα χρησιμοποιούνται ακέραιοι πίνακες, μπορείτε εύκολα να τα μετατρέψετε για να χρησιμοποιούνται με αλφαριθμητικούς πίνακες.

Διαδοχική ή γραμμική αναζήτηση

Αυτός είναι ο αλγόριθμος που χρησιμοποιείται πιο συχνά στην αναζήτηση και συνεπάγεται ν' αρχίσει η αναζήτηση απ' την αρχή της λίστας και να προχωρήσει συγκρίνοντας στη σειρά το κάθε στοιχείο με το «κλειδί» της αναζήτησης. Όταν φτάσει στο τέλος της λίστας και δεν έχει βρεθεί όμοιο στοιχείο, η αναζήτηση θεωρείται πως απέτυχε. Χρησιμοποιήσαμε την απλή αυτή τεχνική στο Πρόγραμμα 4.1 του προηγούμενου κεφαλαίου. Το Πρόγραμμα 5.19 είναι μία επίδειξη της απλής αυτής τεχνικής. Η υπορουτίνα, που αρχίζει απ' τη γραμμή 1000, είναι τόσο απλή που δε χρειάζεται εξήγηση. Οι προηγούμενες γραμμές δημιουργούν έναν πίνακα που περιέχει διαδοχικούς περιττούς αριθμούς. Έτσι, όλοι οι περιττοί αριθμοί της περιοχής που έχουμε θα υπάρχουν στον πίνακα, ενώ όλοι οι άρτιοι θα απουσιάζουν. Τρέξτε το πρόγραμμα με, ας πούμε, 1000 στοιχεία και θα δείτε πόσος πολύς

χρόνος χρειάζεται για να προσδιορίσει ότι ο συγκεκριμένος ακέραιος που εισάγεται στην item% στη γραμμή 100 δεν υπάρχει (ή για να τον βρει, αν υπάρχει) στον πίνακα. Κάντε σύγκριση του χρόνου με τον χρόνο που παίρνει η αναζήτηση με την πιο αποτελεσματική δυαδική μέθοδο που δίνεται στο Πρόγραμμα 5.20.

```
10 REM LINEAR SEARCH DEMONSTRATION
20 CLS
30 INPUT"How many number in list";NUMBER
%
40 DIM A%(NUMBER%)
50 FOR N%=1 TO NUMBER%
60 A%(N%)=N%*2-1
70 PRINT A%(N%)
80 NEXT
90 PRINT
100 INPUT"Search for ";item%
110 START=TIME/300
120 GOSUB 1000
130 T=TIME/300-START
140 PRINT"Serching time=";ROUND(T,2);"Se
conds"
150 INPUT"SERCH AGAIN (Y/N)";K$
160 K$=UPPER$(K$)
170 IF K$="Y" THEN 90
180 END
190 '
200 '
999 REM SEQUENTIAL SEARCH SUBROUTINE
1000 J%=1
1010 WHILE J%<NUMBER% AND item%<>A%(j%)
1020 j%=j%+1
1030 WEND
1040 IF item%=A%(j%) THEN PRINT"Item fou
nd at array position";j% ELSE PRINT"Item
 not found"
1050 RETURN
```

Πρόγραμμα 5.19 Επίδειξη της απλής διαδοχικής αναζήτησης

Η διαδοχική αναζήτηση, αν και είναι σχετικά εύκολο να την κατανοήσει κανείς και να την κάνει πρόγραμμα, είναι προφανώς αργή γιατί, κατά μέσον όρο, θα χρειαστεί να ερευνηθεί το μισό αρχείο πριν βρεθεί το απαιτούμενο δεδομένο. Μ' άλλα λόγια, αν η λίστα την οποία ερευνούμε έχει N στοιχεία, θα χρειαστούν κατά μέσον όρο N/2 αναζητήσεις.

Η δυαδική αναζήτηση

Μια πολύ πιο γρήγορη μέθοδος αναζήτησης ενός πεδίου-κλειδιού, υπό τον όρο ότι οι εγγραφές έχουν πρώτα ταξινομηθεί, είναι η δυαδική αναζήτηση. Με την εξαίρεση της υπορουτίνας στην γραμμή 1000, είναι όμοια στή μορφή με την προηγούμενη μέθοδο.

```
10 REM BINAY SEARCH DEMOSTRATION
20 CLS
30 INPUT"How many numbers in list";NUMBE
RS%
40 DIM A%(NUMBER%)
50 FOR N%=1 TO NUMBER%
60 A%(N%)=N%*2-1
70 PRINT A%(N%)
80 NEXT
90 PRINT
100 INPUT"Search for ";item%
110 START=TIME/300
120 GOSUB 1000
130 T=TIME/300-START
140 PRINT"Searching time=";ROUND(T,2);"S
econds"
150 INPUT"SEARCH AGAIN (Y/N)";K$
160 K$=UPPER$(K$)
170 IF K$="Y" THEN 90
180 END
190 '
200 '
999 REM BINARY SEARCH SUBROUTINE
1000 low%=1:high%=NUMBER%
1010 WHILE high%>low%
1020 mid%=(low%+high%)/2
1030 IF item%>A%(mid%) THEN low%=mid%+1
ELSE high%=mid%
1040 WEND
1050 IF item%=A%(high%) THEN PRINT"Item
found at array position";high% ELSE PRIN
T"Item not found or array not in order"
1060 RETURN
```

Πρόγραμμα 5.20 Επίδειξη της δυαδικής μεθόδου αναζήτησης.

Υποθέτοντας ότι ο πίνακας έχει πρώτα ταξινομηθεί κατ' αύξουσα σειρά, το δεδομένο που βρίσκεται στο μέσον της λίστας είναι το πρώτο που συγκρίνεται με το δεδομένο του οποίου ψάχνουμε το ταίρι. Αν το δεδομένο εί-

ναι μικρότερο απ' το ζητούμενο, η αναζήτηση συνεχίζεται στο πρώτο μισό του πίνακα. Αν είναι μεγαλύτερο, η έρευνα επικεντρώνεται στο δεύτερο μισό του πίνακα. Εντοπίζοντας αυτό το μισό, η διαδικασία συνεχίζεται όπως και πριν ελέγχοντας πρώτα το μεσαίο στοιχείο αυτού του μισού. Τελικά, κόβοντας συνεχώς στη μέση και ελέγχοντας είτε βρίσκουμε το ζητούμενο δεδομένο, είτε διαπιστώνουμε πως είναι ανύπαρκτο στη λίστα. Επιφανειακά, αυτό ίσως φανεί διαδικασία πιο μακρόχρονη από την απλή διαδοχική αναζήτηση, αλλά αυτό συμβαίνει μόνο και μόνο επειδή παίρνει πιο πολλή ώρα να την εξηγήσουμε. Η εξίσωση που μας ενδιαφέρει είναι:

Μέσος όρος συγκρίσεων = $\log_2 n$

όπου n είναι ο συνολικός αριθμός των δεδομένων που πρόκειται να ερευνηθούν.

Πρόκειται για εντυπωσιακό αποτέλεσμα και αξίζει να δώσουμε ένα παράδειγμα, αν μη τι άλλο για να καταδείξουμε την ανωτερότητα της δυαδικής αναζήτησης σε σχέση με την απλή διαδοχική.

Ας υποθέσουμε ότι θέλουμε να εντοπίσουμε ένα συγκεκριμένο στοιχείο από μία λίστα που συνολικά περιέχει 1000 στοιχεία. Θα συγκρίνουμε τις δύο μεθόδους.

(α) Διαδοχική αναζήτηση:

Μέσος όρος συγκρίσεων = n/2 = 1000/2 = 500

(β) Δυαδική αναζήτηση:

Μέσος όρος συγκρίσεων = $\log_2(n) = \log_2(1000) = 10$ (στρογγυλεμένος).

Ακόμα και μ' ένα εκατομμύριο στοιχεία, ο αριθμός των συγκρίσεων δεν είναι παρά 20. Αλλά, για να είμαστε απόλυτα αντικειμενικοί, πρέπει να τονίσουμε για μία ακόμα φορά ότι η δυαδική αναζήτηση μπορεί να πραγματοποιηθεί μόνο σ' ένα αρχείο που έχει προηγουμένως ταξινομηθεί, ενώ η διαδοχική αναζήτηση δεν θέτει καμιά απαίτηση ως προς την τάξη των στοιχείων του αρχείου. Αν ένα αρχείο είναι μικρό σε μέγεθος, μπορεί να μην αξίζει πάντα τον κόπο να το ταξινομήσετε και σίγουρα δεν είναι ιδιαίτερα λογικό να το ταξινομήσετε μόνο και μόνο για να χρησιμοποιήσετε τη δυαδική αναζήτηση. Από την άλλη μεριά, αν ένα αρχείο πρέπει να ταξινομηθεί για άλλους λόγους, θα ήταν ανόητο να μην χρησιμοποιήσετε τη δυαδική μέθοδο. Ένα άλλο σημείο είναι ότι πολλά αρχεία που στηρίζονται στη RAM συχνά ταξινομούνται και φυλάγονται σε κασέτα σε αλφαβητική σειρά έτσι κι αλλιώς.

Περίληψη

1. Η ταξινόμηση με ανταλλαγή είναι απλή αλλά αργή. Ο χρόνος εκτέλε-

σης αυξάνει περίπου ανάλογα με την τρίτη δύναμη του η.

2. Η bubblesort είναι γρηγορότερη απ' την ταξινόμηση με ανταλλαγή εκτός αν ο πίνακας είναι σχεδόν διαταγμένος. Ο χρόνος εκτέλεσης είναι περίπου ανάλογος με το τετράγωνο του η.
3. Εισάγοντας μία «σημαία ανταλλαγής» στη bubblesort ο χρόνος εκτέλεσης μειώνεται για διαταγμένους ή σχεδόν διαταγμένους πίνακες.
4. Η ταξινόμηση με τη μέθοδο του φθίνοντος βήματος μπορεί να χρησιμοποιεί και πάλι τεχνικές της bubblesort αλλά πρώτα κατατέμνει τον πίνακα σε μικρά σύνολα. Ύστερα τα ταξινομημένα σύνολα συγχωνεύονται προοδευτικά σε μεγαλύτερα μέχρι που να μείνει ένα μόνο ταξινομημένο σύνολο.
5. Η εκδοχή 2 της μεθόδου του φθίνοντος βήματος είναι ακόμα πιο αποδοτική απ' την εκδοχή 1.
6. Η Quicksort είναι γενικά γρήγορη, αλλά η επίδοσή της μειώνεται σε διαταγμένους ή σχεδόν διαταγμένους πίνακες.
7. Ο πιο κατάλληλος αλγόριθμος ταξινόμησης για κώδικα μηχανής είναι η απλή ταξινόμηση φθίνοντος βήματος (εκδοχή 1).
8. Η κατανόηση μιας ταξινόμησης με κώδικα μηχανής γίνεται ευκολότερη αν είναι γνωστές οι λεπτομέρειες των αλφαριθμητικών περιγραφέων.
9. Αρχεία μπορούν να αποθηκεύονται σε μορφή δισδιάστατων πινάκων ή, υπό την προϋπόθεση ότι τα αρχεία/εγγραφές είναι όλα καθορισμένου μήκους, σαν ένας απλός αλφαριθμητικός πίνακας.
10. Ένα συγκεκριμένο αρχείο μιας εγγραφής μπορεί να εντοπιστεί είτε με τη μέθοδο της διαδοχικής αναζήτησης ή, αν η εγγραφή είναι ταξινομημένη, με δυαδική αναζήτηση.
11. Η διαδοχική αναζήτηση μπορεί να εφαρμοστεί και σε όχι ταξινομημένο αρχείο.

Ασκήσεις

5.1 Χρησιμοποιώντας την εξίσωση που δίνεται, υπολογίστε (με προσέγγιση εκατομμυρίου) τον αριθμό των συγκρίσεων που χρειάζονται για μία ταξινόμηση με ανταλλαγή αν ο πίνακας περιέχει 500 στοιχεία κατά την ανάστροφη σειρά.

5.2 Σε σχέση με την παραπάνω άσκηση 5.1, αν κάθε κύκλος σύγκρισης παίρνει 1 μιλισεκόντ υπολογίστε τον απαιτούμενο χρόνο ταξινόμησης κατά προσέγγιση ώρας.

5.3 Υπολογίστε τον μέσο αριθμό συγκρίσεων που γίνονται στην ταξινόμηση 1000 στοιχείων ενός πίνακα.

5.4 Βρήτε ένα πιθανό μειονέκτημα της Quicksort.

5.5 Στη Shellsort, Πρόγραμμα 5.5, γιατί προστίθεται στη γραμμή 1020 η σταθερά (2);

Κεφάλαιο Έξι

Έλεγχος γνώσεων

Εισαγωγή

Μερικοί αναγνώστες μπορεί να έχουν τη γνώμη ότι ένα ολόκληρο κεφάλαιο με θέμα τον έλεγχο γνώσεων δεν έχει τη θέση του σ' ένα βιβλίο αφιερωμένο στα συστήματα αρχειοθέτησης. Κι όμως, τα προγράμματα που διευκολύνουν τη σύνθεση ερωτήσεων, η συνακόλουθη αποθήκευσή τους σαν αρχεία δεδομένων και η τελική ανάκλησή τους σε μία μορφή κατάλληλη για να αυτο-ελεγχθεί ο σπουδαστής είναι, ουσιαστικά, ένα σύστημα αρχειοθέτησης που έχει πολλές εφαρμογές. Τα προγράμματα που ακολουθούν ασχολούνται με τη λεγόμενη μέθοδο *πολλαπλής επιλογής* για έλεγχο των γνώσεων.

Η μελέτη οποιουδήποτε γνωστικού αντικειμένου, είτε αυτό είναι η κηπουρική είτε η αστροφυσική, εμπεριέχει την απομνημόνευση ενός αριθμού στοιχείων και κατόπιν την εκμάθηση του πώς αυτά μπαίνουν σε χρήση. Μερικοί μοντέρνοι θεωρητικοί της εκπαίδευσης έχουν την τάση να περιφρονούν τα «ξερά» στοιχεία, με το επιχείρημα ότι αυτά μπορεί πάντα να τα αναζητήσει κανείς σ' ένα κατάλληλο βιβλίο αναφοράς τη στιγμή που θα τα χρειαστεί. Αυτή είναι εξαίρετη συμβουλή, υπό τον όρο να μην την πάρει κανείς και πολύ σοβαρά! Είναι πολύ δύσκολο να εκτιμήσει κάποιος την σχετική αξία δύο αντικρουόμενων επιχειρημάτων αν είναι αναγκασμένος να έχει πάντα δίπλα του την *Εγκυκλοπαίδεια Britannica*. Όπως και να έχει το θέμα, τα περισσότερα μαθήματα οποιουδήποτε κλάδου κορυφώνονται σε κάποιας μορφής εξέταση και προς το παρόν δεν είναι πολλοί οι αρμόδιοι που είναι αρκετά φωτισμένοι ώστε ν' αφήσουν τους σπουδαστές να μπαίνουν στην αίθουσα των εξετάσεων κουβαλώντας σωρούς από βιβλία αναφοράς και άλλο βοηθητικό υλικό. Το πρόβλημα στην εποχή μας είναι ο απίστευτα μεγάλος αριθμός στοιχείων που πρέπει να κουβαλάει κανείς μέσα στο κεφάλι του πριν είναι σε θέση να αρχίσει τη σοβαρή μελέτη του όποιου γνωστικού αντικειμένου.

'Ελεγχος προόδου

Σε κάθε διαδικασία εκμάθησης, απαραίτητη είναι κάποιας μορφής *ανατροφοδότηση (feedback)*. Αυτή μπορεί να είναι είτε ο παραδοσιακός σαρκασμός του διδάσκοντα, ή το σε τακτά χρονικά διαστήματα διαγώνισμα στην αίθουσα διδασκαλίας ή έστω ένας έλεγχος από ένα άλλο μέλος της οικογένειας ή της ομάδας μελέτης. Χωρίς περιοδικούς ελέγχους προόδου είναι δύσκολο να εκτιμηθεί αν υπήρξε κάποια πρόοδος ή όχι. Το ερώτημα που ανακύπτει είναι η μορφή που πρέπει να πάρει αυτός ο έλεγχος προόδου. Αν είναι να γίνει στο σπίτι, η συνηθισμένη πρακτική είναι να ζητήσει ο μαθητής από κάποιον εθελοντή - μέλος της οικογένειας να κάνει το τεστ βάζοντάς του ερωτήσεις που θα τις βρίσκει με τυχαία σειρά στα διδακτικά βιβλία και βλέποντας αν οι απαντήσεις που δίνει ο μαθητής είναι σωστές. Αυτή μέθοδος έχει κάποια αξία αλλά μερικές φορές μπορεί να οδηγήσει σε συγκρούσεις, διότι:

(α) Ο γονιός μπορεί να δυσκολεύεται να βάλει λογικές ερωτήσεις, ακόμα και με τη βοήθεια του διδακτικού βιβλίου.

(β) Η απάντηση που δίνει ο μαθητής μπορεί μεν να είναι σωστή αλλά, επειδή την εκφράζει με λόγια κάπως διαφορετικά απ' αυτά της απάντησης του βιβλίου, ο γονιός τη θεωρεί λάθος (Και οι αντεγκλήσεις που ακολουθούν, συχνά καταλήγουν στη ματαίωση του τεστ!)

(γ) Ο εθελοντής, επειδή έχει το βιβλίο στα χέρια του, μπορεί σιγά-σιγά να αρχίσει να αποκτά ένα υπεροπτικό ύφος που μπορεί να ενοχλήσει τον μαθητή. Βλέπετε, συχνά αποδεχόμαστε με καλή πίστη τις κριτικές όταν έρχονται από κάποιον απέξω, αλλά τις θεωρούμε προσβλητικές όταν γίνονται από ένα μέλος της οικογένειάς μας ή της ομάδας μας.

Τεστ πολλαπλής επιλογής

Ανεξάρτητα αν αυτό είναι δικαιολογημένο ή όχι, οι πιο πολλοί άνθρωποι βλέπουν τον υπολογιστή σαν ένα παντοδύναμο μηχάνημα απαλλαγμένο από προσωπικές κακίες και εντελώς αντικειμενικό σ' ότι αφορά τις συναλλαγές του με τα ανθρώπινα όντα. 'Ετσι, οι έλεγχοι της προόδου που γίνονται μέσω ενός πληκτρολογίου και μιας οθόνης, μπορούν να διεξαχθούν μέσα σε μία ατμόσφαιρα αποφορτισμένη από εντάσεις, η οποία πολύ συμβάλλει στην μάθηση. Αν δώσετε μία λανθασμένη απάντηση, παρ' όλο που βρίσκεστε στο σπίτι σας, κανείς δε θα το μάθει εκτός απ' τον υπολογιστή. Μία κακή τελική βαθμολογία, που εμφανίζεται στην οθόνη μετά το τέλος του τεστ, μπορεί να μείνει μυστικό ανάμεσα σε σας και τον υπολογιστή, και το πιθανότερο είναι να σας δημιουργήσει το ερέθισμα να επαναλάβετε το τεστ και πάλι, και πάλι, μέχρι που να «νικήσετε» τον υπολογιστή. 'Ετσι, η έμφυτη στον άνθρωπο επιθυμία να νικήσει τον υπολογιστή, που τόσο φανερή είναι στους παθιασμένους με τα ηλεκτρονικά παιχνίδια, μπορεί να χρησιμοποιηθεί, έτσι που να φέρνει ωφέλιμο αποτέλεσμα.

Παρ' όλο που υπάρχουν πολλοί τρόποι για να καταρτιστεί ένα τεστ

προόδου στον υπολογιστή, η μέθοδος της πολλαπλής επιλογής από τέσσερις απαντήσεις σε μιαν ερώτηση είναι η πιο εύλογη. Αυτό το είδος τεστ χρησιμοποιήθηκε για πρώτη φορά κατά τον Δεύτερο Παγκόσμιο Πόλεμο. Πρόσφερνε μια γρήγορη μέθοδο για να ελεγχθεί η πρόοδος στρατιωτικού προσωπικού που έκανε επιμορφωτικά σεμινάρια. Τα παραδοσιακά διαγωνίσματα απαιτούσαν να υπάρχει μία ομάδα βαθμολογητών με υψηλή κατάρτιση και θα διαρκούσαν πάρα πολύ. Αντίθετα, οι κόλλες των διαγωνισμάτων πολλαπλής επιλογής μπορούσαν να βαθμολογηθούν από ανειδίκευτο προσωπικό, που απλώς θα έβαζε ένα στένσιλ πάνω στο φύλλο των απαντήσεων. Για κάποιο διάστημα μετά τον πόλεμο, τα εκπαιδευτικά ιδρύματα περιφρονούσαν το σύστημα αυτό, θεωρώντας το σαν ένα φτηνό «τέχνασμα» κατάλληλο μόνο για να βαθμολογούνται χαμηλού επιπέδου σπουδαστές όταν κάνουν προγύμναση. (Το κατεστημένο πάντα είχε την τάση να θεωρεί την «προγύμναση» κατώτερη της «εκπαίδευσης»). Ωστόσο, εξαιτίας της τεχνολογικής προόδου κατά τα μεταπολεμικά χρόνια, η στάση απέναντι στα τεστ πολλαπλής επιλογής άλλαξε και, τελικά, ακόμα και τα πανεπιστήμια άρχισαν να το αποδέχονται σαν ένα χρήσιμο συμπλήρωμα στα παραδοσιακά διαγωνίσματα. Το Ανοιχτό Πανεπιστήμιο ευνοεί ιδιαίτερα αυτή τη μορφή τεστ.

Μηχανισμός του τεστ πολλαπλής επιλογής

Το τεστ αποτελείται από μία σειρά ερωτήσεων, που η καθεμιά έχει τουλάχιστον τέσσερις απαντήσεις, A, B, C και D. Ο διαγωνιζόμενος βάζει ένα σταυρό δίπλα στην απάντηση που θεωρεί σαν «καλύτερη». Στις πιο περίπλοκες ερωτήσεις, είναι πιθανόν τρεις από τις ερωτήσεις να φαίνονται επιφανειακά σωστές, αλλά μόνο η μία είναι πραγματικά σωστή απ' όλες τις απόψεις. Το παρακάτω παράδειγμα είναι από τα απλά:

Ερώτηση: Ποιά είναι η πρωτεύουσα της Ουγγαρίας;
Απάντηση A: Η Σόφια
Απάντηση B: Η Βουδαπέστη
Απάντηση C: Η Βαρσοβία
Απάντηση D: Το Βουκουρέστι.

Δεν είναι καλή μέθοδος να υπάρχει μία απάντηση παράλογη, γιατί αυτό στην πράξη περιορίζει την επιλογή στις τρεις απαντήσεις. Θα πρέπει να θυμόμαστε ότι κατά τη διάρκεια της διαδικασίας των απαντήσεων είναι αναπόφευκτο ότι ο μαθητής μερικά πράγματα θα τα μαντέψει χωρίς να τα ξέρει σίγουρα. Στο παραπάνω παράδειγμα, και οι τέσσερις προτεινόμενες απαντήσεις έχουν ανατολικοευρωπαϊκό «χρώμα» κι έτσι είναι όλες πιθανές, αλλά αν βάζαμε τη Νέα Υόρκη σαν προτεινόμενη απάντηση, αυτή θα απορριπτόταν σχεδόν σίγουρα. Την εποχή που πρωτάρχισαν τα τεστ πολλαπλής επιλογής αιτία πολλών επικρίσεων ήταν η τάση που είχαν αυτοί που τα έφτιαχναν να καταφεύγουν σε φτηνά παιχνίδια με τις λέξεις. Προσπαθούσαν να διατυπώσουν τις τέσσερις απαντήσεις με τέτοιο τρόπο που να παραπλανήσουν τον καημένο τον μαθητή και να τον κάνουν να βάλει το σταυρό δίπλα στη

λαθεμένη απάντηση.

Η μέθοδος της πολλαπλής επιλογής χρησιμοποιείται εκτεταμένα για να γίνεται «κοσκίνισμα» αυτών που υποβάλλουν αιτήσεις για να προσληφθούν σε μία θέση. Η διεξαγωγή προσωπικών συνεντεύξεων είναι δαπανηρή δουλειά, ειδικά αν η συγκεκριμένη θέση απαιτεί τεχνικές ή διοικητικές γνώσεις. Ένα προκαταρκτικό τεστ πολλαπλής επιλογής θα αποκλείσει τη μεγάλη μάζα των ανεπαρκών υποψηφίων. Εξαιτίας αυτού, είναι καλό για όλους, και για παιδιά και για μεγάλους, να εξοικειωθούν μ' αυτό το σύστημα. Άλλο είναι να ξέρεις καλά κάποιο αντικείμενο και εντελώς άλλο να βρεθείς μπροστά σ' ένα ερωτηματολόγιο με μορφή πολλαπλής επιλογής για πρώτη φορά στη ζωή σου και μάλιστα κάτω από συνθήκες έντασης και τρακ.

Η παρουσίαση στην οθόνη του υπολογιστή

Ο Amstrad είναι ιδανικός για ερωτήσεις πολλαπλής επιλογής γιατί διαθέτει 80 στήλες στην Επιλογή 2 και η οθόνη του είναι κατάλληλη γι' αυτή τη διαδικασία. Υπάρχουν κι άλλοι δημοφιλείς υπολογιστές που διαθέτουν 80 στήλες αλλά η παρουσίαση σε οθόνη τηλεόρασης, ειδικά αν είναι έγχρωμης συσκευής, είναι σχεδόν αδύνατο να διαβαστεί. Η παρουσίαση των 80 στηλών στον Amstrad είναι *άριστη στη μονόχρωμη οθόνη* και εντελώς ευανάγνωστη ακόμα και σε χρωματιστή. Φυσικά, δεν είναι απόλυτα απαραίτητο να διαλέξετε την παρουσίαση των 80 στηλών αλλά αν προσπαθήσετε να το κάνετε με 40 μόνο στήλες θα υπάρξουν προβλήματα. Είναι αυτονόητο ότι η οθόνη πρέπει να είναι ικανή να εμφανίζει την ερώτηση, τις τέσσερις απαντήσεις και κάποια μορφή προτρεπτικού μηνύματος. Με 40 μόνο στήλες και, ας πούμε, 25 διαθέσιμες γραμμές, είναι πολύ δύσκολο να συνθέσει κανείς ερωτήσεις και απαντήσεις που να κεντρίζουν τη σκέψη και αρκετά συνοπτικές ώστε να χωράνε σε μία σελίδα οθόνης. Με 25 γραμμές των 80 στηλών η καθεμιά, υπάρχει χώρος για τουλάχιστον 256 χαρακτήρες για την ερώτηση και 256 για καθεμιά απ' τις τέσσερις απαντήσεις, που χωράει άνετα ακόμα και τις πιο λεπτομερείς και ερευνητικές ερωτήσεις.

Μέθοδοι προγραμματισμού

Υπάρχει ποικιλία στους τρόπους με τους οποίους μπορούν να προγραμματιστούν οι ερωτήσεις του τεστ. Μία απλή μορφή προγράμματος μπορεί να επινοηθεί, στην οποία οι ερωτήσεις και οι απαντήσεις περιέχονται σε προτάσεις DATA. Και πραγματικά, προγράμματα που βασίζονται σ' αυτή την αρχή έχουν εμφανιστεί σε ένα-δύο μηνιαία περιοδικά για προγραμματιστές, αλλά, εκτός απ' την απλότητά τους, λίγα μπορεί να επαινέσει κανείς σ' αυτά τα προγράμματα. Τα ελαττώματα είναι φανερά:

(α) Αυτός που διαγωνίζεται μπορεί εύκολα να κάνει LIST στο πρόγραμμα και να εξετάσει τα DATA ώστε να βρει τη σωστή απάντηση.

(β) Αν αυτός που καταρτίζει το τεστ δεν έχει κάποια εμπειρία με τους υπολογιστές θα έχει δυσκολίες να καταστρώσει τις εντολές DATA.

Το πρώτο που πρέπει να σκεφτόμαστε είναι να σχεδιάζουμε το πρόγραμμα με διάθεση φιλικότητας προς τον χρήστη. Πρέπει να είναι εύκολη και η εισαγωγή των ερωτήσεων και η διεξαγωγή του τεστ. Ελπίζουμε ότι τα προγράμματα που ακολουθούν θα ενθαρρύνουν τα μέλη της οικογένειας να κατασκευάσουν ζωντανές ερωτήσεις ή ακόμα και να τροποποιήσουν τα προγράμματα έτσι που να τους ταιριάζουν πιο πολύ.

Η πιο αποδοτική μέθοδος είναι να «σπάσουμε» τη δουλειά σε δύο κομμάτια, ένα πρόγραμμα για τη σύνταξη των ερωτήσεων και ένα ξεχωριστό πρόγραμμα για την απάντηση των ερωτήσεων. Θα ονομάσουμε τα προγράμματα αυτά «MC Prep» και «MC Test» αντίστοιχα. Παρακάτω στο κεφάλαιο αυτό δίνονται οι λίστες δύο τέτοιων προγραμμάτων μαζί με λειτουργικές λεπτομέρειες. Στο μεταξύ, είναι βολικό να μελετήσουμε μερικές απ' τις επιλογές που έχουμε συμπεριλάβει και να υποδείξουμε πως μπορούν κάποιες απ' αυτές να τροποποιηθούν ή να διευρυνθούν ώστε να καλύπτουν τυχόν προσωπικές απαιτήσεις.

Προετοιμασία αρχείου πολλαπλής επιλογής (Πρόγραμμα 6.1)

Το τελικό αποτέλεσμα του προγράμματος είναι η παραγωγή μιας *κασέτας δεδομένων* που να περιέχει τις ερωτήσεις και τις απαντήσεις μαζί με κάποιες οδηγίες σχετικές με το τεστ. Απαραίτητο είναι να έχετε μία άγραφη κασέτα έτοιμη στο κασετόφωνο πριν τρέξετε το πρόγραμμα. Η ταινία των δεδομένων πρέπει να σωθεί για συνακόλουθη χρήση στο δεύτερο πρόγραμμα, το MC Test (Πρόγραμμα 6.2).

Λειτουργικές λεπτομέρειες.

Το πρόγραμμα καθοδηγείται από μενού. Όταν αρχίζει να τρέχει το πρόγραμμα, εμφανίζεται η σελίδα του μενού. Το επάνω μισό δίνει ορισμένες βασικές πληροφορίες και το κάτω μισό δείχνει τις διαθέσιμες επιλογές. Πριν εισαχθούν οι βασικές αυτές οδηγίες, η σελίδα του μενού έχει την πιο κάτω εμφάνιση:

```
        Προετοιμασία
     πολλαπλής επιλογής

Αντικείμενο         :
Τεστ                :
Συντάκτης           :

Ημερομηνία          :
Πηγές               :
Χρόνος (σε λεπτά)   :
Μέγεθος αρχείου     :  Ø ερωτήσεις
```

1 Προετοιμασία αρχείου 2 Φόρτωμα αρχείου 3 Σώσιμο αρχείου 4 Εμφάνιση αρχείου 5 Προσθήκη ερωτήσεων 6 Τροποποίηση ερωτήσεων 7 Διαγραφή ερωτήσεων 8 Έξοδος απ' το αρχείο Διαλέξτε επιλογή:

Επιλογή 1: Προετοιμασία του αρχείου

Αυτή η επιλογή θα πρέπει να χρησιμοποιείται όταν προετοιμάζουμε μία καινούργια ομάδα ερωτήσεων για να δοθούν οι βασικές πληροφορίες απ' το συντάκτη των ερωτήσεων. Μέχρι να γίνει αυτό, το μέγεθος των ερωτήσεων (ο αριθμός των ερωτήσεων) παραμένει 0. Για τον μέγιστο αριθμό των ερωτήσεων το όριο είναι εκατό, αλλά δεν απαιτείται να εισαχθεί το μέγεθος του αρχείου μια και ανανεώνεται αυτόματα καθώς προστίθενται ερωτήσεις. Το τελευταίο στοιχείο που απαιτείται είναι ο χρόνος (σε λεπτά) που ο συντάκτης των ερωτήσεων θεωρεί επαρκή και λογικό για την απάντησή τους. Αφού εισαχθούν αυτές οι βασικές πληροφορίες, η σελίδα του μενού αντικαθίσταται από την οθόνη «Add Question» (Προσθήκη ερωτήσεων). Αυτή αρχίζει με την ακόλουθη πληροφοριακή επικεφαλίδα:

MC Preparation Bytes free: xxxxx Question number1 Προετοιμασία πολλ. επιλογής Ελεύθερα bytes: xxxxx Ερώτηση αριθμός 1

Το πρώτο σήμα αναμονής του υπολογιστή ζητά την ερώτηση, τα επόμενα τέσσερα τις απαντήσεις. Το τελευταίο προτρεπτικό σήμα ρωτάει ποιά από τις τέσσερις απαντήσεις A, B, C ή D είναι η σωστή. Καθώς εισάγεται μία ερώτηση, η πληροφοριακή επικεφαλίδα ενημερώνεται ανάλογα. Είναι σημαντικό να ξέρει ο χρήστης πόσα byte του απομένουν για να μην εξαντλήσει τη μνήμη. Παρ' όλο που υπάρχει άνω όριο για τις ερωτήσεις, το 100, είναι πιθανό να εξαντληθεί η μνήμη πριν φτάσουμε στο όριο αυτό αν κάθε ερώτηση έχει άφθονο κείμενο. Πολύ σημαντικό είναι να θυμάστε ότι αν μία ερώτηση ή απάντηση παίρνει πάνω από μία γραμμή, *δεν πρέπει να χρησιμοποιηθεί το πλήκτρο ENTER για να περάσουμε στην επόμενη γραμμή*, γιατί αυτό θα σταματήσει την καταχώρηση. Το όριο που βάζει το λειτουργικό σύστημα του Amstrad στον αριθμό των χαρακτήρων μιας αλφαριθμητικής μεταβλητής που περιέχεται στη RAM είναι το 255, αλλά το πρόγραμμα βάζει όριο στους 254 χαρακτήρες γιατί διαπιστώσαμε ότι μερικές φορές τα αρχεία

δεδομένων σε κασέτες απορρίπτουν τους 255 χαρακτήρες σε μία αλφαριθμητική. Καμιά ερώτηση ή απάντηση δεν πρέπει να υπερβεί το όριο αυτό· αν το υπερβεί, θα ακουστεί ένα προειδοποιητικό ηχητικό σήμα και οι πλεονάζοντες χαρακτήρες θ' απορριφθούν. Ωστόσο, είναι τελείως απίθανο να υπάρξει ερώτηση, και πολύ περισσότερο απάντηση, που θα καταναλώσει πάνω από 254 χαρακτήρες. Οι περισσότερες ερωτήσεις σπάνια θα ξεπερνάνε τους 80 χαρακτήρες. Για να κάνουμε έναν συντηρητικό πρόχειρο υπολογισμό, ας υποθέσουμε ότι κάθε ερώτηση και κάθε απάντηση απαιτεί 100 χαρακτήρες, που δίνουν σύνολο 500 bytes, δηλαδή περίπου μισό K μνήμης ανά ερώτηση. 'Αρα, σ' ένα αρχείο υπάρχει αρκετός χώρος για 50 ερωτήσεις, μια και μόνο 25K θα χρησιμοποιηθούν, αφήνοντας πάλι ένα άνετο περιθώριο.

Αφού εισαχθεί η πρώτη ερώτηση και οι απαντήσεις, το πρόγραμμα επιστρέφει στο μενού. Κάθε αρχείο που τυχόν υπάρχει στη RAM καταστρέφεται όταν επιλέξουμε την επιλογή 1.

Επιλογή 5: Προσθήκη ερωτήσεων

Απ' τη στιγμή που το αρχείο έχει προετοιμαστεί και έχει εισαχθεί η ερώτηση 1, χρησιμοποιείται αυτή εδώ η επιλογή κάθε φορά που πρόκειται να προσθέσουμε στο αρχείο μία επιπλέον ερώτηση. Ο αύξων αριθμός της ερώτησης, ο αριθμός των bytes που απομένουν και τα σήματα αναμονής που ζητούν ερώτηση και απαντήσεις είναι όπως περιγράφονται στην Επιλογή 1, πιο πάνω. Αφού εισαχθούν η ερώτηση, οι τέσσερις απαντήσεις και η σωστή απάντηση, επιστρέφουμε πάλι στο μενού, οπότε πρέπει να επιλέξουμε πάλι την Επιλογή 5 για να προχωρήσουμε σε νέα ερώτηση.

Επιλογή 3: Σώσιμο του αρχείου

Δεν είναι υποχρεωτικό να κάθεστε μπρος στο πληκτρολόγιο ώρες ατέλειωτες μέχρι που να περάσετε όλες τις ερωτήσεις. Σε όποια ερώτηση κι αν έχετε φτάσει, διαλέγοντας την Επιλογή 3 μπορείτε να σώσετε ένα όχι απόλυτα ολοκληρωμένο αρχείο σε κασέτα. Θυμηθείτε ότι πρέπει να βγάλετε την κασέτα του προγράμματος και να την αντικαταστήσετε με μίαν άλλη τελείως άγραφη ή με κάποια κασέτα που την έχετε πριν γυρίσει στο άγραφο τμήμα της. Το μόνο σήμα αναμονής (prompt) που παίρνετε, αφορά το όνομα του αρχείου ακολουθούμενο απ' τις συνηθισμένες οδηγίες του συστήματος να πιέσετε το REC και το PLAY. Το πρόγραμμα καθορίζει την ταχύτητα εγγραφής στα 2000 baud αλλά αν είστε δύσπιστος μπορείτε πάντα να περάσετε στην ταχύτητα που εξ ορισμού χρησιμοποιεί το σύστημα, τα 1000 baud. Το ίδιο ισχύει και για την έκδοση δισκέτας.

Επιλογή 2: Φόρτωμα του αρχείου

Υποθέτοντας ότι η κασέτα έχει τυλιχτεί στη σωστή θέση και ότι η απάντησή σας στο σήμα αναμονής «Δώστε όνομα αρχείου» έγινε αποδεκτή, το

αρχείο φορτώνεται. Η θέση στον υπολογιστή είναι τώρα ίδια ακριβώς μ' αυτήν που ήταν πριν σωθεί το συγκεκριμένο αρχείο. Για παράδειγμα, αν το αρχείο έχει σωθεί στη φάση που έχει ολοκληρωθεί και η ερώτηση 15, τώρα είναι έτοιμο να δεχτεί τις ερωτήσεις από 16 και πέρα. Τα αρχεία δισκέτας φορτώνονται με τον ίδιο τρόπο.

Επιλογή 4: Εμφάνιση του αρχείου

Η επιλογή αυτή σας επιτρέπει να *δείτε μία - μία* όλες τις ερωτήσεις του αρχείου για να τις εξετάσετε. Το πρώτο σήμα αναμονής ζητά από σας τον αριθμό της ερώτησης, πράγμα που σημαίνει ότι μπορείτε ν' αρχίσετε όχι απ' την αρχή αλλά από όποιον αριθμό δώσετε. Αν δώσετε ανύπαρκτο ή απαράδεκτο αριθμό θα πάρετε μήνυμα λάθους της μορφής «Εύρος ερωτήσεων = x ως y». Ανεξάρτητα από ποιά ερώτηση αρχίσατε την εξέταση, μπορείτε να φέρετε στην οθόνη άλλες ερωτήσεις χρησιμοποιώντας τα πλήκτρα του αριστερού ή του δεξιού δρομέα. Ο δεξιός δρομέας προχωράει στην ερώτηση με τον αμέσως μεγαλύτερο αύξοντα αριθμό και ο αριστερός στην ερώτηση με τον αμέσως μικρότερο αύξοντα αριθμό. Η ενέργεια των δρομέων προχωράει κυκλικά πέρα απ' τα δύο άκρα κι έτσι αν, για παράδειγμα, η τελευταία ερώτηση του αρχείου έχει αριθμό 47 και πατήσουμε το πλήκτρο του δεξιού δρομέα θα πάρουμε την ερώτηση αριθμός 1. Για να επιστρέψετε στο μενού, πατήστε το πλήκτρο του κενού.

Επιλογή 6: Τροποποίηση ερωτήσεων

Φυσικά, είναι πολύ βασικό να μπορεί κανείς να τροποποιήσει είτε την ερώτηση είτε μία ή περισσότερες απ' τις απαντήσεις. Ακόμα κι αν οι αρχικές ερωτήσεις έχουν πληκτρολογηθεί σωστά, σε οποιοδήποτε τεστ πολλαπλής επιλογής υπάρχει πάντα μεγάλη πιθανότητα να βρεθεί κάποιος που να αμφισβητήσει την αξία μιας ερώτησης. Είτε η ερώτηση έχει διατυπωθεί παραπλανητικά, είτε —συχνότερα απ' ότι φαίνεται— δύο απαντήσεις στην ίδια ερώτηση κρίνονται εξίσου σωστές. Γι' αυτό το λόγο είναι αναγκαίο να υπάρχει δυνατότητα τροποποίησης ώστε να μπορεί ο χρήστης ν' αλλάξει οποιοδήποτε ξεχωριστό κομμάτι κάποιας ερώτησης. Θα πρέπει επίσης να υπάρχει η δυνατότητα ν' αλλάξουν οι βασικές οδηγίες στο μενού. Για παράδειγμα, ο χρόνος που κρίνεται επιτρεπτός για την απάντηση στο τεστ. Στην Επιλογή 6 έχει δοθεί ιδιαίτερη φροντίδα ώστε να υπάρχει ξεχωριστή πρόσβαση σε κάθε τμήμα του προγράμματος και για να γίνει η διαδικασία της τροποποίησης όσο γίνεται πιο άκοπη.

Την πρώτη φορά που διαλέγετε την επιλογή αυτή, στην οθόνη εμφανίζονται τα εξής:

```
Για να τροποποιήσετε το Μενού
     επιλέξτε ερώτηση 0
Δώστε τον αριθμό της ερώτησης
```

Έτσι, αν χρειάζεται τροποποίηση μόνο στις βασικές οδηγίες της αρχής, θα πρέπει να διαλέξετε την ερώτηση 0. Ενώ για να τροποποιήσετε κάποια ερώτηση, πληκτρολογείτε τον αντίστοιχο αριθμό.

Τότε στην οθόνη εμφανίζεται:

```
Ποιό τμήμα θέλετε να τροποποιήσετε;
    (1) Ερώτηση
    (2) Απάντηση Α
    (3) Απάντηση Β
    (4) Απάντηση C
    (5) Απάντηση D
    (6) Σωστή απάντηση Α, Β, C ή D;
    Διαλέξτε επιλογή:
```

Υποθέτοντας ότι διαλέξατε το 5 (Απάντηση D) στην οθόνη τώρα εμφανίζεται:

Μπορείτε να χρησιμοποιήσετε όσο θέλετε απ' το ΥΠΑΡΧΟΝ ΚΕΙΜΕΝΟ με τα πλήκτρα CURSOR/COPY. Για να επιστρέψετε στο μενού, πληκτρολογήστε EXIT. Τώρα τροποποιείτε το πεδίο με επικεφαλίδα D.
(Εδώ εμφανίζεται το υπάρχον κείμενο)
(ΔΩΣΤΕ ΕΔΩ ΤΟ ΝΕΟ ΚΕΙΜΕΝΟ)

Μόλις τροποποιηθεί η γραμμή όπου υπάρχει το πρόβλημα, εμφανίζεται γι' άλλη μια φορά για έναν τελευταίο έλεγχο. Χρησιμοποιήστε τα πλήκτρα SHIFT και CURSOR/COPY για να ξαναχρησιμοποιήσετε όσο απ' το υπάρχον κείμενο θέλετε. Για να επιστρέψετε στο μενού, πληκτρολογήστε EXIT.

Επιλογή 7: Απαλοιφή ερώτησης

Αυτή η επιλογή ζητάει απλώς τον αριθμό της ερώτησης που πρόκειται να σβηστεί και κλείνει τα κενά στο αρχείο. Αν η επιλογή διαλέχτηκε κατά λάθος, υπάρχει η δυνατότητα να επιστρέψετε απ' ευθείας στο μενού, διαλέγοντας την ερώτηση 0. Αυτή η επιλογή αναγκάζει επίσης σε συλλογή αχρήστων τους πλεονάζοντες αλφαριθμητικούς χαρακτήρες που υπάρχουν στη μνήμη. Αν στο αρχείο υπάρχουν πολλές ερωτήσεις, αυτή η εργασία πιθανόν να πάρει κάμποση ώρα.

Επιλογή 8: Έξοδος από το πρόγραμμα.

Είναι σημαντικό να χρησιμοποιείτε αυτήν την επιλογή, και όχι το πλήκτρο ESC, όταν έχετε τελειώσει το πρόγραμμα, γιατί αποκαθιστά τις εξ ορισμού τιμές στην κανονική τους κατάσταση.

Ανάλυση του Προγράμματος 6.1

Τα σχόλια που θ' ακολουθήσουν αναφέρονται στη λίστα του Προγράμματος 6.1.

```
10 REM MULTIPLE CHOICE FILE PREPARATION
20 OPENOUT"buffer"
30 MEMORY &9B7E
40 CLOSEOUT
50 flag%=0
60 DIM A$(100,5),H$(5),P$(5)
70 H$(0)="QUESTION"
80 H$(1)="ANSWER A"
90 H$(2)="ANSWER B"
100 H$(3)="ANSWER C"
110 H$(4)="ANSWER D"
120 H$(5)="CORRECT answer (A,B,C,D or D)
"
130 P$(0)="Subject"
140 P$(1)="Test"
150 P$(2)="Author"
160 P$(3)="Date"
170 P$(4)="Ref"
180 P$(5)="Time (minutes)"
190 BORDER 0
200 MODE 1
210 SEL%=0
220 WHILE SEL%<1 OR SEL%>8
230 GOSUB 2110:GOSUB 2190
240 LOCATE 1,14
250 PRINT"(1)   PREPARE file"
260 PRINT"(2)   LOAD file"
270 PRINT"(3)   SAVE file"
280 PRINT"(4)   VIEW file"
290 PRINT"(5)   ADD Question"
300 PRINT"(6)   MODIFY Question"
310 PRINT"(7)   DELETE question"
320 PRINT"(8)   EXIT program"
330 LOCATE 1,24
340 PRINT"Select option :";
```

```
350 K$=INKEY$:IF K$="" THEN 350
360 SEL%=VAL(K$)
370 WEND
380 CLS
390 IF SEL%>2 AND SIZE%=0 AND SEL%<>8 TH
EN PRINT"No File loaded":GOSUB 1800:GOTO
 200
400 IF SEL%<3 AND flag%=1 THEN GOSUB 185
0
410 ON SEL% GOSUB 470,570,690,810,960,11
20,1460,430
420 GOTO 200
430 SPEED WRITE 0
440 END
450 '
460 REM PREPARATE NEW FILE
470 SIZE%=0:K%=22
480 FOR C%=0 TO 5
490 PRINT"Enter ";P$(C%)
500 GOSUB 1950:A$(0,C%)=UPPER$(K$)
510 NEXT
520 GOSUB 960
530 flag%=1
540 RETURN
550 '
560 REM LOAD FILE
570 GOSUB 2050
580 OPENIN filename$
590 INPUT#9,SIZE%
600 FOR Q%=0 TO SIZE%
610 FOR C%=0 TO 5
620 INPUT#9,A$(Q%,C%)
630 NEXT:NEXT
640 CLOSEIN
650 flag%=1
660 RETURN
670 '
680 REM SAVE FILE
690 GOSUB 2050
700 SPEED WRITE 1
710 OPENOUT filename$
720 PRINT#9,SIZE%
730 FOR Q%=0 TO SIZE%
740 FOR C%=0 TO 5
750 PRINT#9,A$(Q%,C%)
```

```
760 NEXT:NEXT
770 CLOSEOUT
780 RETURN
790 '
800 REM VIEW FILE
810 LOW%=1:GOSUB 1730
820 IF Q%<1 THEN Q%=SIZE%
830 IF Q%>SIZE% THEN Q%=1
840 GOSUB 1620
850 FOR C%=0 TO 5
860 PRINT H$(C%);" : ";USING "&";A$(Q%,C
%)
870 NEXT
880 WHILE INKEY(47)<>0
890 CALL &BB18:'KM WAIT KEY
900 IF INKEY(8)=0 THEN Q%=Q%-1:GOTO 820
910 IF INKEY(1)=0 THEN Q%=Q%+1:GOTO 820
920 WEND
930 RETURN
940 '
950 REM ADD QUESTION
960 IF SIZE%>=100 OR FRE(0)<2000 THEN PR
INT"FILE FULL":GOSUB 1800:GOTO 1090
970 SIZE%=SIZE%+1
980 Q%=SIZE%:GOSUB 1620
990 K%=254
1000 FOR C%=0 TO 5
1010 PRINT"Enter ";H$(C%)" : ";
1020 GOSUB 1950:A$(Q%,C%)=K$
1030 X%=POS(#0):Y%=VPOS(#0)
1040 LOCATE 48,2
1050 PRINT USING "#####";FRE(0)
1060 LOCATE X%,Y%
1070 NEXT
1080 A$(Q%,5)=UPPER$(A$(Q%,5))
1090 RETURN
1100 '
1110 REM MODIFY QUESTION
1120 PRINT"To Modify MENU PANEL select q
uestion 0":LOW%=0:GOSUB 1730
1130 SEL%=0
1140 WHILE SEL%<1 OR SEL%>6
1150 CLS:LOCATE 2,5:PRINT"MODIFY which p
art of QUESTION ";Q%
1160 LOCATE 1,10
```

```
1170 FOR C%=1 TO 6
1180 PRINT"(";C%;")"    ";
1190 IF Q%=0 THEN PRINT P$(C%-1)
1200 IF Q%>0 THEN PRINT H$(C%-1)
1210 NEXT
1220 LOCATE 1,21:PRINT"Select option"
1230 K$=INKEY$:IF K$="" THEN 1230
1240 SEL%=VAL(K$)
1250 WEND
1260 K$="":WHILE K$<>"EXIT" AND K$<>"exi
t"
1270 GOSUB 1620
1280 PRINT"You can utilise any of the EX
ISTING TEXT with the CURSOR/COPY keys"
1290 PRINT
1300 PRINT"Type EXIT to regain the main
MENU"
1310 PRINT
1320 PRINT"You are MODIFYING the field l
abelled : ";
1330 IF Q%=0 THEN PRINT P$(SEL%-1)
1340 IF Q%>0 THEN PRINT H$(SEL%-1)
1350 PRINT:PRINT:GOSUB 2010
1360 PRINT A$(Q%,SEL%-1)
1370 GOSUB 2010:LOCATE 1,VPOS(#0)+4:GOSU
B 2010
1380 LOCATE 1,VPOS(#0)-5
1390 INPUT K$
1400 IF Q%=0 OR SEL%=6 THEN K$=UPPER$(K$
)
1410 IF K$<>"EXIT" AND K$<>"exit" THEN A
$(Q%,SEL%-1)=K$
1420 WEND
1430 RETURN
1440 '
1450 REM DELETE QUESTION
1460 PRINT"The DELETE option has been se
lected"
1470 PRINT"To return to MENU select Ques
tion 0"
1480 LOW%=0:GOSUB 1730
1490 IF Q%=0 THEN 1590
1500 PRINT"Wait"
1510 WHILE Q%<SIZE%
1520 FOR C%=0 TO 5
```

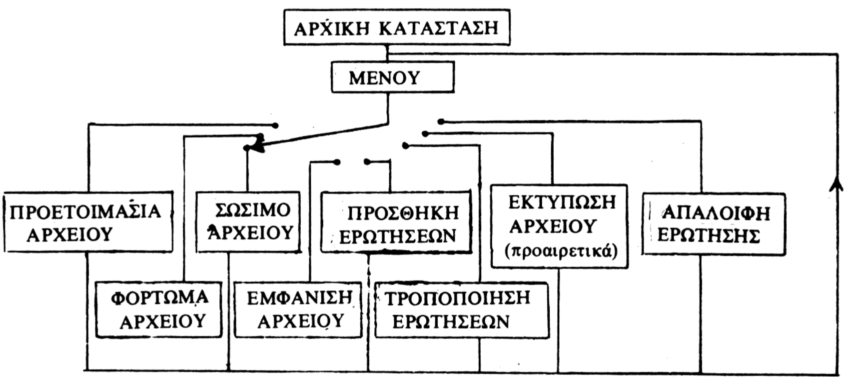
```
1530 A$(Q%,C%)=A$(Q%+1,C%)
1540 NEXT
1550 Q%=Q%+1
1560 WEND
1570 SIZE%=SIZE%-1
1580 PRINT FRE("")
1590 RETURN
1600 '
1610 REM DISPLAY HEADER
1620 MODE 2
1630 GOSUB 2010
1640 PRINT"MULTIPLE CHOICE PREPARATION"
1650 LOCATE 35,2
1660 PRINT"Bytes free :";FRE(0)
1670 LOCATE 64,2
1680 PRINT"QUESTION :";Q%
1690 GOSUB 2010
1700 PRINT
1710 RETURN
1720
1730 PRINT
1740 REM GET QUESTION NUMBER
1750 INPUT"Give Question Number ";Q%
1760 IF Q%<LOW% OR Q%>SIZE% THEN PRINT"R
ANGE ("LOW%;"to";SIZE%")":GOTO 1750
1770 RETURN
1780 '
1790 REM PRESS ANY KEY
1800 PRINT"PRESS any KEY to CONTINUE"
1810 CALL &BB18:'KM WAIT KEY
1820 RETURN
1830 '
1840 REM BELT &BRACES
1850 PRINT"LOADED FILE AT RISK"
1860 PRINT"Do you wish to RESELECT MENU
(Y/N)";
1870 K$=INKEY$
1880 K$=UPPER$(K$)
1890 IF K$<>"Y" AND K$<>"N" THEN 1870
1900 IF K$="Y" THEN SEL%=0
1910 IF K$="N" THEN flag%=0
1920 RETURN
1930 '
1940 REM GET LINE INPUT
1950 LINE INPUT K$
```

```
1960 IF K$="" THEN 1950
1970 IF LEN(K$)>K% THEN K$=LEFT$(K$,K%)
1980 RETURN
1990 '
2000 REM PRINT A LINE
2010 PRINT STRING$(80,CHR$(154));
2020 RETURN
2030 '
2040 REM GET FILENAME
2050 CLS:PRINT"ENTER filename"
2060 K%=16:GOSUB 1950
2070 filename$=K$
2080 RETURN
2090 '
2100 REM DISPLAY HEADER 2
2110 CLS
2120 PRINT
2130 INK 2,6
2140 PAPER 2
2150 PRINT" MULTIPLE CHOICE PREPARATION"
2160 PAPER 0
2170 RETURN
2180 '
2190 REM DISPLAY PANEL
2200 PRINT
2210 PRINT STRING$(40,CHR$(154));
2220 FOR C%=0 TO 5
2230 PRINT TAB(2) P$(C%) TAB (16) ": "A$
(0,C%)
2240 NEXT
2250 PRINT" File size      :";SIZE%;"QUES
TIONS"
2260 PRINT STRING$(40,CHR$(154));
2270 RETURN
```

Πρόγραμμα 6.1. Προετοιμασία του αρχείου πολλαπλής επιλογής.

Το πρόγραμμα φυσικά διαιρείται σε δύο τμήματα. Το πρώτο, που φτάνει ως το END της γραμμής 440, είναι το καθαυτό «πρόγραμμα». Οι υπόλοιπες γραμμές σχηματίζουν ένα «οπλοστάσιο» από υπορουτίνες. Οι βασικές υπορουτίνες, που φέρνουν σε πέρας τις διάφορες λειτουργίες του μενού, καλούνται από το κυρίως πρόγραμμα. Αλλά κι αυτές με τη σειρά τους μπορούν να καλούν μικρότερες υπορουτίνες οι οποίες εκτελούν όχι εξειδικευμένες αλλά μάλλον γενικού χαρακτήρα λειτουργίες. Τα ονόματα των μεταβλητών και οι αντίστοιχες λειτουργίες τους δίνονται παρακάτω:

A$(Q%, C%) = βασικός πίνακας που περιέχει ερωτήσεις και απαντήσεις, με διαστάσεις A$(100,5).
H$(0-5) = αμετάβλητα σήματα αναμονής για τις ερωτήσεις και τις απαντήσεις.
P$(0-5) = αμετάβλητα σήματα αναμονής για τις βασικές πληροφορίες του μενού.
SEL% = ο αριθμός της επιλογής που διαλέχτηκε.
SIZE% = ο μεγαλύτερος αριθμός ερώτησης αυτή τη στιγμή στο αρχείο.
K$ = γενικού χαρακτήρα συνολική μεταβλητή
filename$ = τρέχον όνομα αρχείου.
Q% = αριθμός ερώτησης
C% = δείκτης είτε για την ίδια την ερώτηση, είτε για τις τέσσερις απαντήσεις ή τη σωστή απάντηση.
flag% = κατάσταση του αρχείου: 0 = αρχείο όχι μόνιμο 1 = αρχείο μόνιμο.



Σχ. 6.1 Δομικό διάγραμμα του Προγράμματος 6.1

Η διαδρομή των υπορουτινών.

Οι βασικές υπορουτίνες για την κάθε επιλογή καλούνται από την εντολή ON SEL% GOSUB στη γραμμή 500. Η καθεμιά επιστρέφει (RETURN) στην αρχή του μενού στη γραμμή 200, μέσω του GOTO της γραμμής 420. Η συνολική δομή φαίνεται στο Σχήμα 6.1

Κλήσεις των υπορουτινών

Μερικές υπορουτίνες καλούν άλλες υπορουτίνες. Ο πίνακας 6.1 δείχνει ποιές απ' αυτές καλούνται σε κάθε επιλογή.

Το πρώτο τμήμα ξεκινάει με μία ρουτίνα που διατηρεί μία σταθερή προσωρινή μνήμη μεγέθους 4K στην κασέτα. Σ' ένα πρόγραμμα τέτοιου τύπου, η μέθοδος αυτή είναι προτιμότερη από την συνηθισμένη δυναμική κατανο-

Πίνακας 6.1. Κλήσεις υπορουτινών

Προετοιμασία ερώτησης	Προετοιμασία νέου αρχείου
	Εμφάνισε δεδομένα ανά γραμμή
	Πρόσθεσε ερώτηση
Φόρτωμα αρχείου	Διάβασε όνομα αρχείου
Σώσιμο αρχείου	Διάβασε όνομα αρχείου
Εμφάνιση αρχείου	Διάβασε αριθμό ερώτησης
	Εμφάνισε επικεφαλίδα
Προσθήκη ερώτησης	Εμφάνισε δεδομένα ανά γραμμή
Τροποποίηση ερώτησης	Διάβασε αριθμό ερώτησης
	Εμφάνισε επικεφαλίδα
	Τύπωσε μια γραμμή
Απαλοιφή ερώτησης	Διάβασε αριθμό ερώτησης

μή προσωρινής μνήμης που κάνει το HIMEM να κινείται πάνω - κάτω. Αυτό μπορεί να είναι μία μακρόχρονη και επαναληπτική διαδικασία, γιατί όλες οι αλφαριθμητικές που είναι στο «σωρό» πρέπει να αποσυρθούν στη συλλογή αχρήστων, να μεταφερθούν έξω για να δημιουργήσουν χώρο για την προσωρινή μνήμη της κασέτας και, ύστερα από ένα φόρτωμα ή σώσιμο, να επανατοποθετηθούν. (Κοιτάξτε την παράγραφο που κάνει λόγο για τα μπλοκ δεδομένων κασέτας στο Κεφάλαιο 1).

Για τη σελίδα του μενού χρησιμοποιείται οθόνη 40 στηλών, αλλά για την προετοιμασία και παρουσίαση των ερωτήσεων η υπορουτίνα εμφάνισης επικεφαλίδας βάζει την οθόνη στις 80 στήλες.

Πρόγραμμα του τεστ πολλαπλής επιλογής

Ο σκοπός του πιο πάνω προγράμματος ήταν η δημιουργία μιας ΚΑΣΕΤΑΣ Η ΔΙΣΚΕΤΑΣ ΤΟΥ ΤΕΣΤ (αρχείου δεδομένων) που περιέχει τις ερωτήσεις. Αυτό που θα δούμε πιο κάτω, το Πρόγραμμα 6.2, την χρησιμοποιεί για να παρουσιάζει τις ερωτήσεις, να δέχεται τις απαντήσεις και να εμφανίζει την τελική βαθμολογία.

Λειτουργία του προγράμματος

Στο πρόγραμμα εμφυτεύτηκε σ' ένα βαθμό η τυπικότητα του σχολείου, κι έτσι δίνεται σε κάποιον «επιβλέποντα» η εξουσία να εμποδίζει τον διαγωνιζόμενο να κάνει πάνω από μία προσπάθεια! Έτσι, ο υπεύθυνος θα πρέπει να χειριστεί το πρώτο μέρος του προγράμματος ως εξής:

(1) Φορτώνει το Πρόγραμμα 6.2

(2) Απομακρύνει απ' τον οδηγό κασέτας την κασέτα του προγράμματος και κρατάει την ΚΑΣΕΤΑ ΤΟΥ ΤΕΣΤ (αρχείο δεδομένων) στο χέρι του.
(3) Τρέχει (RUN) το Πρόγραμμα 6.2.
(Σημ. το ίδιο ισχύει αν υπάρχει σύστημα δίσκων μόνο που τα πράγματα είναι ευκολότερα μια και δεν χρειάζονται ξετυλίγματα κλπ.).

Στην οθόνη θα εμφανιστούν τα εξής:

ΤΕΣΤ ΠΟΛΛΑΠΛΗΣ ΕΠΙΛΟΓΗΣ
Τοποθέτησε την ΚΑΣΕΤΑ
του ΤΕΣΤ στο μηχάνημα
ΤΥΛΙΞΕ την κασέτα να είναι
έτοιμη να φορτώσει ΔΕΔΟΜΕΝΑ
Πληκτρολόγησε το όνομα του αρχείου

Όταν το αρχείο έχει φορτωθεί, το επόμενο σήμα αναμονής είναι:

ΑΠΟΜΝΗΜΟΝΕΥΣΕ και ΠΛΗΚΤΡΟΛΟΓΗΣΕ
κάποιον ΕΞΑΨΗΦΙΟ κωδικό

Απ' τη στιγμή που δοθεί αυτός ο κωδικός, ο διαγωνιζόμενος δεν θα μπορεί να ξανατρέξει το πρόγραμμα, γιατί το πλήκτρο ESC έχει γίνει ανενεργό. Αν γίνει απόπειρα από το hardware, το πρόγραμμα χάνεται έτσι κι αλλιώς, και μόνο ο επιβλέπων έχει την κασέτα με το πρόγραμμα που χρειάζεται για ένα ξανατρέξιμο. Το τελευταίο σήμα αναμονής της οθόνης, σ' ότι αφορά τον επιβλέποντα, είναι το όνομα του διαγωνιζόμενου. Τότε ο υπολογιστής είναι έτοιμος για να αρχίσει το τεστ.

Οι ενέργειες του μαθητή

Η πρώτη οθόνη δίνει το Αντικείμενο, το Τεστ, τον Συντάκτη, την Ημερομηνία, τις Πηγές και τον Χρόνο που επιτρέπεται (σε λεπτά). Ακολουθούν οδηγίες ως προς το ποιά πλήκτρα πρέπει να πιέσει ο μαθητής για να απαντήσει. Αρχίζει το τεστ και το ρολόι αρχίζει να αφαιρεί χρόνο απ' τη στιγμή που ο μαθητής πατήσει κάποιο πλήκτρο και εμφανίζεται η ακόλουθη εικόνα (παραδείγματος χάρη) στην οθόνη:

Πολλαπλή Επιλογή: Χρόνος 01:15:05 Ερώτηση: 1

Ερώτηση: Αν N ακέραιος,
ποιά απ' τις παρακάτω παραστάσεις
είναι σίγουρα ΠΕΡΙΤΤΟΣ
ακέραιος;
Απάντηση A: 2N
Απάντηση B: 2N+1
Απάντηση C:N+1
Απάντηση D: N/3

Δεν έχει δοθεί ΑΠΑΝΤΗΣΗ Πίεσε A, B, C ή D για να δώσεις ΑΠΑΝΤΗΣΗ ή να ΑΛΛΑΞΕΙΣ ΑΠΑΝΤΗΣΗ.

Το τεστ αρχίζει στην ερώτηση υπ' αριθμόν 1, αλλά η επόμενη ερώτηση δεν ακολουθεί αυτόματα μετά την απάντηση της προηγούμενης. Για να προχωρήσει ο διαγωνιζόμενος προς τα μπρος ή προς τα πίσω στην επόμενη ερώτηση, πρέπει να χρησιμοποιήσει τα πλήκτρα του δεξιού ή του αριστερού δρομέα. Η απάντηση στην κάθε ερώτηση δίνεται πληκτρολογώντας A, B, C ή D είτε σε πεζά είτε σε κεφαλαία. Ο μαθητής μπορεί όποτε θέλει ν' αλλάξει γνώμη και να διορθώσει την απάντησή του. Στην αρχή εμφανίζεται το μήνυμα «Δεν έχει δοθεί ΑΠΑΝΤΗΣΗ», αλλά απ' τη στιγμή που δοθεί κάποια απάντηση, το μήνυμα αλλάζει και γίνεται: «Η ΑΠΑΝΤΗΣΗ σας είναι:» ακολουθούμενο απ' το γράμμα που δόθηκε τελευταία φορά σαν απάντηση. Καθώς το τεστ προχωράει, ο χρόνος που απομένει σε ώρες, λεπτά και δευτερόλεπτα είναι πάντα ορατός στην κορυφή της οθόνης μαζί με τον αριθμό της ερώτησης που εκείνη τη στιγμή απαντιέται. Όταν τελικά το ρολόι φτάσει στο μηδέν το τεστ σταματάει, ανεξάρτητα απ' τον αριθμό των ερωτήσεων που έχουν απαντηθεί. Το μέτρημα των τελευταίων 10 δευτερολέπτων συνοδεύεται από βόμβους που χρησιμεύουν σαν προειδοποίηση για τον μαθητή. Αν ο σπουδαστής τελειώσει το τεστ πριν τελειώσει ο χρόνος, μπορεί, αν θέλει, να πατήσει το πλήκτρο ESC.

Φτάνοντας στο χρόνο μηδέν, ή με το πάτημα του πλήκτρου ESC, στην οθόνη εμφανίζεται:

ΤΕΛΟΣ ΤΟΥ ΤΕΣΤ
Πατήστε οποιοδήποτε πλήκτρο για να πάρετε αποτελέσματα

Τότε εμφανίζονται στην οθόνη τα αποτελέσματα του τεστ στην μορφή που βλέπετε στο παρακάτω παράδειγμα:

ΤΕΣΤ ΠΟΛΛΑΠΛΗΣ ΕΠΙΛΟΓΗΣ

Αντικείμενο	Μαθηματικά
Τεστ	Σειρές Taylor
Συντάκτης	Ντ. Τζ. Στέφενσον
Ημερομηνία	25 Οκτωβρίου 1984
Πηγές	M/S2/302
Χρόνος (σε λεπτά)	37
Μέγεθος αρχείου	50 ερωτήσεις

ΠΙΝΑΚΑΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ:	Χ. Γουώρτον
Συνολικές ερωτήσεις	50
Ερωτήσεις που απαντήθηκαν	40
Σωστές απαντήσεις	20
Λάθος απαντήσεις	20

Ποσοστό επιτυχίας	40%
Βαθμολογία	D(Βάση)
Δώστε τον κωδικό (CODE) για να πάρετε το μενού	

Επιστροφή στο μενού

Έχοντας τελειώσει το τεστ, είναι διαθέσιμες τέσσερις επιλογές:

(α) Να επιτραπεί στον ίδιο σπουδαστή να ξαναδώσει το τεστ
(β) Να επιτραπεί σε άλλον σπουδαστή να δώσει το τεστ.
(γ) Να φορτωθεί ένα διαφορετικό σετ ερωτήσεων.
(δ) Να βγείτε απ' το πρόγραμμα.

Αν υποθέσουμε ότι το τεστ διεξάγεται με όλους τους τύπους, ο μαθητής που μόλις τέλειωσε την εξέτασή του δεν θα έχει πρόσβαση στο μενού, κι έτσι τον έλεγχο θα τον έχει ο επιβλέπων γιατί μόνο αυτός θα ξέρει τον κωδικό που είχε δοθεί στην αρχή του τεστ.

Κάθε που εισάγεται ένας χαρακτήρας του κωδικού, ακούγεται ένας οξύς τόνος. Για λόγους συγχρονισμού επαναλαμβάνεται ένας χαμηλός τόνος με την εισαγωγή του κάθε έκτου χαρακτήρα του κωδικού. Αν δοθεί ο σωστός κωδικός, θα παρουσιαστεί το μενού. Αν όχι, θα ακουστεί ο χαμηλός τόνος σαν σήμα που δηλώνει ότι δόθηκε λάθος κωδικός και πρέπει να ξαναδοθεί απ' την αρχή. Η σελίδα του μενού παρουσιάζεται ως εξής:

1 Φορτώστε νέο αρχείο ερωτήσεων 2 Επανάληψη του τεστ (ο νέος διαγωνιζόμενος) 3 Επανάληψη του τεστ (ο ίδιος διαγωνιζόμενος) 4 Έξοδος από το πρόγραμμα (Ενεργοποίηση του πλήκτρου ESC)
Διαλέξτε αριθμό επιλογής

Ανεπίσημο τεστ

Παρ' όλο που ως τώρα έχει προϋποτεθεί η παρουσία ενός «επιβλέπο-

ντα», το τεστ μπορεί εύκολα να πραγματοποιηθεί και ανεπίσημα. Ο διαγωνιζόμενος μπορεί να πληκτρολογήσει τον δικό του κωδικό και να επαναλάβει όσες φορές θέλει το τεστ για να βελτιώσει τη βαθμολογία του. Σ' ένα πιο φιλελεύθερο περιβάλλον, το πρόγραμμα μπορεί ν' αποτελέσει μία νεωτεριστική δύναμη που να παρακινεί τους μαθητές για διαρκή αυτοβελτίωση. Βέβαια, υπάρχει μία δυσκολία: πρέπει να υπάρχει κάποιος που να μπορεί κατ' αρχήν να φτιάχνει τις ερωτήσεις και να ξέρει πως χρησιμοποιούνται οι δυνατότητες του Προγράμματος 6.1 πριν χρησιμοποιηθεί το Πρόγραμμα 6.2. Υπάρχουν δύο λύσεις:

(1) Τα σχολεία, ή πιο συχνά τα τεχνικά και επαγγελματικά ιδρύματα και Πανεπιστήμια, συχνά δίνουν πρόθυμα τα θέματα των εξετάσεων του προηγούμενου χρόνου. Πολλά απ' αυτά είναι σε μορφή πολλαπλής επιλογής αν και κανονικά δεν δίνονται και οι απαντήσεις. Παρ' όλ' αυτά, ίσως υπάρχει κάποιος συγγενής ή φίλος που ξέρει κάποιες από τις απαντήσεις. Αλλά κι αν δεν υπάρχει, οι ίδιες οι ερωτήσεις μπορούν ν' αποτελέσουν πρότυπο για απλούστερες ιδέες.

(2) Αν και ίσως φανεί παράλογο, ο μαθητής θα μπορούσε να φτιάξει τις δικές του ερωτήσεις! Δεν υπάρχει καλύτερη μέθοδος για να μάθεις ένα αντικείμενο από το να προσπαθήσεις να το διδάξεις εσύ ο ίδιος. Και ένα σημαντικό τμήμα της διδασκαλίας είναι η δημιουργία ερωτήσεων ελέγχου. Ο μαθητής μπορεί, με τη βοήθεια διδακτικών βιβλίων, να μάθει πάρα πολλά καταρτίζοντας ένα σύνολο από ερωτήσεις και απαντήσεις γιατί μία τέτοια εργασία τον παρακινεί να ερευνά το θέμα. Εδώ μπορεί κάποιος να διαφωνήσει λέγοντας ότι ο μαθητής που καταρτίζει το τεστ λίγο θα ωφεληθεί αν αργότερα εξεταστεί σ' αυτό μια και θα ξέρει ήδη τις απαντήσεις. Η αντίρρηση ισχύει μόνο αν η εξέταση γίνεται μία - δυο μέρες ύστερα απ' τη δημιουργία του τεστ. Τα αποτελέσματα του ίδιου τεστ αν το δώσουμε μία βδομάδα αργότερα, που η μνήμη αρχίζει να εξασθενεί, μπορεί ν' αποδειχθούν μεγάλη έκπληξη όσο και απογοήτευση. Έχοντας αυτό κατά νου, οι μαθητές μπορούν να προετοιμάζουν αρχεία με ερωτήσεις και να τα φυλάνε για να τα χρησιμοποιήσουν αργότερα στις επαναλήψεις τους.

Ανάλυση του Προγράμματος 6.2

Τα παρακάτω πρέπει να διαβαστούν σε συνδυασμό με τη λίστα του Προγράμματος 6.2 και το δομικό διάγραμμα που δίνεται στο Σχήμα 6.2.

```
10 REM MULTIPLE CHOICE TEST PROGRAM
20 OPENOUT"buffer"
30 MEMORY &9B7E
40 CLOSEOUT
50 BORDER 0
60 DIM A$(100,5),H$(5),P$(5),ANS$(100)
70 flag%=0:lastT%=0
```

```
80 H$(0)="QUESTION"
90 H$(1)="ANSWER A"
100 H$(2)="ANSWER B"
110 H$(3)="ANSWER C"
120 H$(4)="ANSWER D"
130 H$(5)="Your RESPONSE stored as : "
140 P$(0)="Subject"
150 P$(1)="Test"
160 P$(2)="Author"
170 P$(3)="Date"
180 P$(4)="Ref"
190 P$(5)="Time (minutes)"
200 MODE 1
210 GOSUB 1480:LOCATE 1,5
220 PRINT"Place TEST CASSETTE in the mac
hine"
230 PRINT"REWIND tape ready to LOAD DATA
"
240 PRINT:GOSUB 830
250 PRINT:PRINT"MEMORISE and ENTER any 6
 DIGIT code"
260 INPUT code$
270 IF LEN(code$)=0 OR LEN(code$)>6 THEN
 250
280 CALL &BB03:'KM RESET
290 GOSUB 1480:LOCATE 1,5
300 PRINT"Enter testee's NAME (Max 22 Ch
aracters)"
310 LINE INPUT name$
320 IF name$="" THEN 300
330 IF LEN(name$)>22 THEN name$=LEFT$(na
me$,22)
340 name$=UPPER$(name$)
350 GOSUB 730
360 GOSUB 950
370 CALL &BB18:'KM WAIT KEY
380 FINISH=TIME/300+VAL(A$(0,5))*60
390 GOSUB 1080
400 SOUND 1,36,20,15
410 MODE 1:LOCATE 16,12
420 PRINT"TEST OVER":PRINT
430 LOCATE 5,14
440 PRINT"PRESS any KEY to DISPLAY RESUL
TS"
450 CALL &BB18:'KM WAIT KEY
```

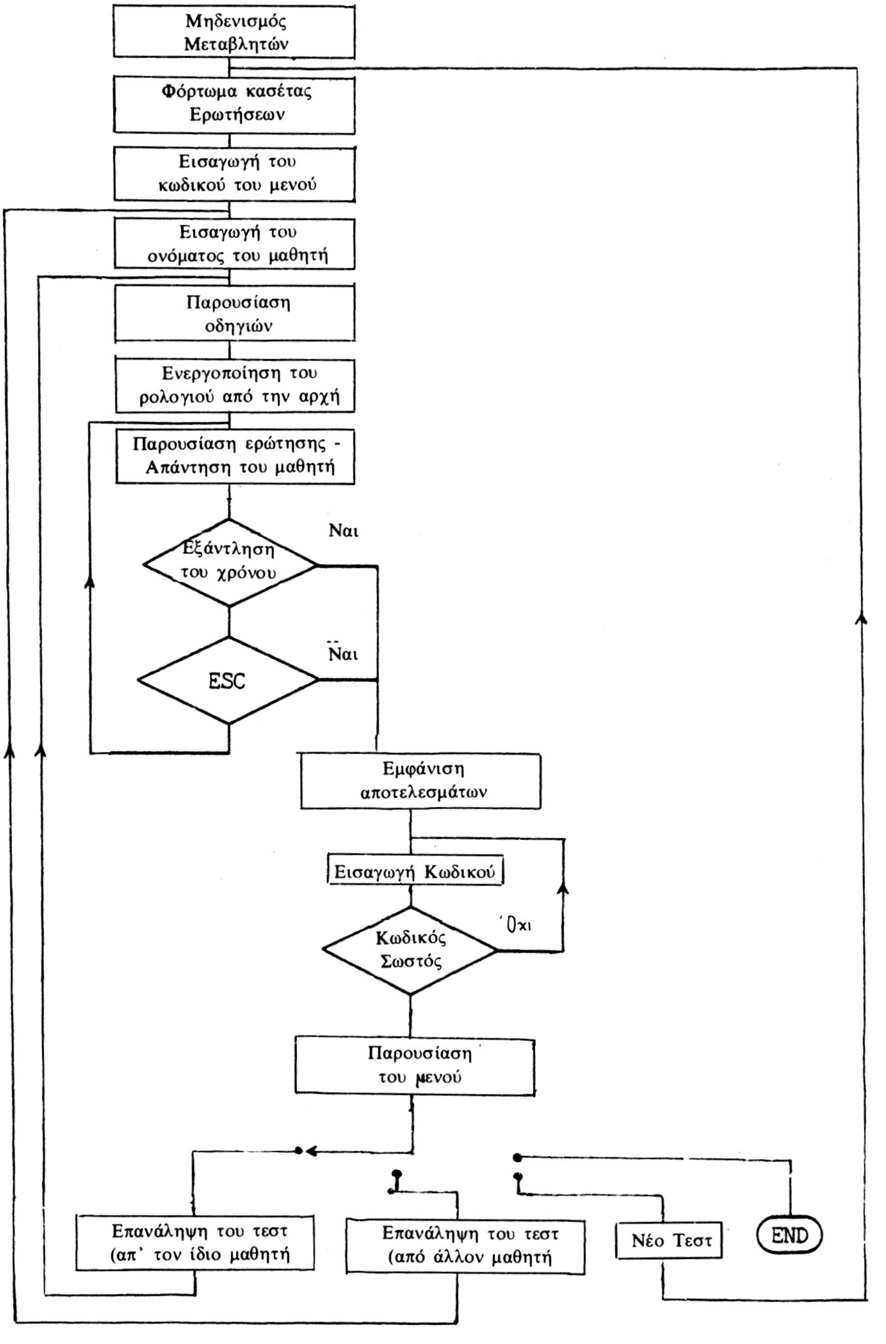
```
460 GOSUB 730
470 GOSUB 1710
480 LOCATE 1,23
490 PRINT STRING$(40,CHR$(154));
500 PRINT"Enter code to OBTAIN MENU"
510 PRINT STRING$(40,CHR$(154));
520 try$="":SOUND 1,758,10,15
530 FOR CHAR%=1 TO 6
540 K$=INKEY$:IF K$="" THEN 540
550 SOUND 1,18,5,15:try$=try$+K$
560 NEXT
570 IF try$<>code$ THEN 520
580 flag%=0:FOR Q%=1 TO SIZE%:ANS$(Q%)="
":NEXT
590 GOSUB 1480:LOCATE 1,10
600 PRINT"  (1) Load new QUESTION FILE"
610 PRINT"  (2) RESIT test (new testee)"
620 PRINT"  (3) RESIT test (same testee)
"
630 PRINT"  (4) EXIT program (Enable ESC
APE key)"
640 LOCATE 2,23
650 PRINT"Select option :"
660 K$=INKEY$:IF K$="" THEN 660
670 SEL%=VAL(K$)
680 IF SEL%<1 OR SEL%>4 THEN 590
690 ON SEL% GOTO 210,290,350,700
700 CLS:END
710 '
720 REM DISPLAY PANEL
730 MODE 1:GOSUB 1480
740 PRINT STRING$(40,CHR$(154));
750 FOR C%=0 TO 5
760 PRINT TAB(2) P$(C%) TAB(16) ": "A$(0
,C%)
770 NEXT
780 PRINT" File size      :";SIZE%"QUESTI
ONS"
790 PRINT STRING$(40,CHR$(154));
800 RETURN
810 '
820 REM LOAD FILE
830 PRINT"Enter filename :";
840 LINE INPUT filename$
850 OPENIN filename$
```

```
860 INPUT#9,SIZE%
870 FOR Q%=0 TO SIZE%
880 FOR C%=0 TO 5
890 INPUT#9,A$(Q%,C%)
900 NEXT:NEXT
910 CLOSEIN
920 RETURN
930 '
940 REM INSTRUCTIONS
950 LOCATE 1,14
960 PRINT"STEP trough questions (wrap a
round) by"
970 PRINT"pressing one of the following
keys :"
980 PRINT"RIGHT arrow key increments QUE
STION No."
990 PRINT"LEFT arrow key decrements QUES
TION No."
1000 PRINT:PRINT STRING$(40,CHR$(154));
1010 PRINT"To END TEST at any time PRESS
 ESC key"
1020 PRINT STRING$(40,CHR$(154));
1030 PRINT
1040 PRINT"To START the TEST press any k
ey"
1050 RETURN
1060 '
1070 REM DISPLAY & ANSWER QUESTIONS
1080 Q%=1
1090 WHILE flag%=0
1100 GOSUB 1350
1110 FOR C%=0 TO 4
1120 PRINT H$(C%);" : ";USING "&";A$(Q%,
C%)
1130 NEXT
1140 LOCATE 1,24:GOSUB 1440
1150 WHILE flag%=0 AND INKEY(8)<>0 AND I
NKEY(1)<>0
1160 LOCATE 1,25
1170 IF ans$(Q%)="" THEN PRINT"No RESPON
SE recorded" ELSE PRINT H$(5);ans$(Q%)
1180 LOCATE 37,25
1190 PRINT"Press A,B,C or D to ANSWER or
 CHANGE ANSWER"
1200 K$=INKEY$
```

```
1210 IF INKEY(66)=0 THEN flag%=1
1220 GOSUB 1580
1230 IF k$="" AND flag%=0 THEN 1200
1240 K$=UPPER$(K$)
1250 IF K$>="A" AND K$<="D" THEN ANS$(Q%
)=K$
1260 WEND
1270 IF INKEY(8)=0 THEN Q%=Q%-1
1280 IF INKEY(1)=0 THEN Q%=Q%+1
1290 IF Q%<1 THEN Q%=SIZE%
1300 IF Q%>SIZE% THEN Q%=1
1310 WEND
1320 RETURN
1330 '
1340 REM DISLAY HEADER 1
1350 MODE 2
1360 GOSUB 1440
1370 PRINT"MULTIPLE CHOICE TEST"
1380 LOCATE 30,2
1390 PRINT"Time left  "
1400 LOCATE 64,2
1410 PRINT"question :";Q%
1420 GOSUB 1440
1430 RETURN
1440 PRINT STRING$(80,CHR$(154));
1450 RETURN
1460 '
1470 REM DISPLAY HEADER 2
1480 CLS
1490 LOCATE 1,2
1500 INK 2,6
1510 PAPER 2
1520 PRINT" MULTIPLE CHOICE TESTING "
1530 PAPER 0
1540 PEN 1
1550 RETURN
1560 '
1570 REM TRACK AND DISPLAY TIME LEFT
1580 T%=FINISH-TIME/300
1590 S%=T% MOD 60
1600 M%=T%/60
1610 H%=M%/60
1620 M%=M% MOD 60
1630 LOCATE 41,2
1640 PRINT USING "## : ## : ##";H%,M%,S%
```

```
1650 IF T%<=10 AND lastT%>T% THEN SOUND
1,36,5,15
1660 IF T%<=0 THEN flag%=1
1670 lastT%=T%
1680 RETURN
1690 '
1700 REM PROCESS AND DISPLAY RESULTS
1710 SCORE%=0:PASS%=0
1720 FOR Q%=1 TO SIZE%
1730 IF ANS$(Q%)=A$(Q%,5) THEN SCORE%=SC
ORE%+1
1740 IF ANS$(Q%)="" THEN PASS%=PASS%+1
1750 NEXT
1760 PRINT"RESULTS TABLE  :  ";name$
1770 PRINT STRING$(40,CHR$(154));:PRINT
1780 PRINT"TOTAL Questions      :";SIZE%
1790 PRINT"Questions ATTEMPTED :";SIZE%-
PASS%
1800 PRINT"CORRECT answers      :";SCORE%
1810 PRINT"INCORRECT answers    :";SIZE%-
PASS%-SCORE%
1820 PERCENT=SCORE%/SIZE%*100
1830 PRINT"PERCENTAGE score     :";ROUND(
PERCENT,2);"%"
1840 PRINT:PRINT"Performance GRADE    : "
;
1850 IF PERCENT>=90 THEN PRINT"A (EXCELL
ENT!)"
1860 IF PERCENT>=70 AND PERCENT<90 THEN
PRINT"B (GOOD PASS)"
1870 IF PERCENT>=50 AND PERCENT<70 THEN
PRINT"C (CLEAR PASS)"
1880 IF PERCENT>=40 AND PERCENT<50 THEN
PRINT"D (PASS)"
1890 IF PERCENT<40 AND PERCENT>=30 THEN
PRINT"E (FAIL)"
1900 IF PERCENT<30 THEN PRINT"F (BAD FAI
L)"
1910 RETURN
```

Πρόγραμμα 6.2 Τεστ πολλαπλής επιλογής.



Σχήμα 6.2 Κατασκευή διαγράμματος για το πρόγραμμα 6.2

Τα ονόματα των μεταβλητών και οι αντίστοιχες λειτουργίες τους είναι ως εξής:

A$(Q%, C%) = βασικός πίνακας που περιέχει ερωτήσεις και απαντήσεις, με διαστάσεις A$(100,5).
H$(0-5) = αμετάβλητα σήματα αναμονής για τις ερωτήσεις και τις απαντήσεις.
P$(0-5) = αμετάβλητα σήματα αναμονής για τις βασικές πληροφορίες/επικεφαλίδες.
SEL% = αριθμός της επιλογής που διαλέχτηκε.
SIZE% = η ερώτηση που έχει υψηλότερο αριθμό.
K$ = συνολική μεταβλητή γενικής χρήσεως.
filename$ = τρέχον όνομα αρχείου
Q% = αριθμός ερώτησης
C% = δείκτης για την ερώτηση αυτή καθαυτή, τις τέσσερις απαντήσεις και τη σωστή απάντηση.
flag% = κατάσταση του τεστ: flag% = 0 τεστ συνεχίζεται. flag% = 1 το τεστ τελείωσε.
code$ = κωδικός 6 χαρακτήρων για επιστροφή στο μενού.
M% = λεπτά
S% = δευτερόλεπτα
H% = ώρες
T% = τικ του ρολογιού

Οι κλήσεις που κάνουν οι υπορουτίνες φαίνονται στον Πίνακα 6.2.

Πίνακας 6.2 Κλήσεις από τις υπορουτίνες

Παρουσίαση και απάντηση ερωτήσεων	Εμφάνιση επικεφαλίδας 1 Τύπωμα της επόμενης γραμμής
Υπολογισμός και παρουσίαση του χρόνου που απομένει	Όχι άλλες κλήσεις
Φόρτωμα αρχείου	Όχι άλλες κλήσεις
Εμφάνιση μενού	Επικεφαλίδα 2
Οδηγίες	Όχι άλλες κλήσεις
Εμφάνιση επικεφαλίδας 1	Τύπωμα της επόμενης γραμμής
Εμφάνιση επικεφαλίδας 2	Όχι άλλες κλήσεις
Επεξεργασία και παρουσίαση αποτελεσμάτων	Όχι άλλες κλήσεις

Αυτό καθαυτό το πρόγραμμα φτάνει ως το END στη γραμμή 700, οι υπόλοιπες γραμμές αποτελούνται από υπορουτίνες.

Οι περισσότερες υπορουτίνες είναι αρκετά απλά δομημένες και, εκτός από μερικά ονόματα μεταβλητών, είναι παρόμοιες στη μορφή μ' αυτές που συζητήσαμε σε προηγούμενα προγράμματα. Η υπορουτίνα στη γραμμή 1580 προϋποθέτει ότι ξέρετε τη συνάρτηση MOD, γιατί τη χρησιμοποιεί για να χωρίζει τον χρόνο σε ώρες, λεπτά και δευτερόλεπτα. Η παρουσίαση του

χρόνου στην οθόνη σε ομάδες των δύο ψηφίων επιτυγχάνεται χρησιμοποιώντας την εντολή PRINT USING της γραμμής 1640.

Από το Σχήμα 6.2 θα πρέπει να γίνεται φανερό πώς αλληλοσυνδέονται οι υπορουτίνες.

Περίληψη

1. Ένα τεστ πολλαπλής επιλογής αποτελείται από ερωτήσεις που η καθεμιά συνοδεύεται από τέσσερις απαντήσεις, που μόνο η μια τους θεωρείται σωστή.
2. Είναι κακή πρακτική να παίζει κανείς σ' αυτά τα τεστ με τις λέξεις.
3. Για την παρουσίαση της ερώτησης και των τεσσάρων απαντήσεων είναι αναγκαία οθόνη των 80 στηλών.
4. Απαιτούνται δύο προγράμματα, ένα για την προετοιμασία των ερωτήσεων και ένα για την απάντησή τους.
5. Και τα δύο προγράμματα περιλαμβάνουν κασέτες ή δισκέτες δεδομένων.
6. Το Πρόγραμμα 6.1 πρέπει να σώσει μία κασέτα ή δισκέτα δεδομένων που να περιέχει τις ερωτήσεις. Το Πρόγραμμα 6.2 φορτώνει μία κασέτα δεδομένων πριν αρχίσει το τεστ.
7. Όταν προετοιμάζετε ερωτήσεις ή απαντήσεις που παίρνουν πάνω από μία γραμμή, να χρησιμοποιείτε τη μπάρα διαστήματος (και όχι το πλήκτρο ENTER) για ν' αλλάξετε γραμμή.
8. Όταν προετοιμάζετε τις ερωτήσεις, να προσέχετε την ένδειξη που δείχνει τα bytes που απομένουν.
9. Η προσθήκη κάθε ερώτησης γίνεται μέσω της σελίδας του μενού.
10. Το πρόγραμμα 6.1 παράγει μία κασέτα ή δισκέτα δεδομένων του τύπου των 2000 baud.
11. Κατά την προετοιμασία, ο χρήστης μπορεί να δει οποιαδήποτε ερώτηση δίνοντας τον αντίστοιχο αριθμό.
12. Μπορεί κανείς να σβήσει ερωτήσεις, αλλά οι αριθμοί των ερωτήσεων που ακολουθούν θα μειωθούν κατά ένα για να κρατηθεί η αριθμητική συνέχεια.
13. Όταν κάποιος διαγωνίζεται στο τεστ χρησιμοποιώντας το Πρόγραμμα 6.2, το τελικό μενού μπορεί να προστατευτεί με μία λέξη-κωδικό που εισάγει ο εξεταστής.
14. Ενώ το τεστ προχωράει, ο χρόνος που απομένει εμφανίζεται στο άνω μέρος της οθόνης.
15. Όταν μένουν μόνο 10 δευτερόλεπτα για να τελειώσει ο χρόνος του τεστ, δίνεται ηχητικό σήμα προειδοποίησης.
16. Κατά τη διάρκεια του τεστ, το πλήκτρο ESC έχει απενεργοποιηθεί.

Άσκηση

Χρησιμοποιώντας το Πρόγραμμα 6.1 και με τη βοήθεια ενός λεξικού, συνθέστε ένα σύνολο πενήντα ερωτήσεων πάνω στην ορθογραφία. Για παράδειγμα:

Ερώτηση: Ποιά είναι η σωστή ορθογραφία της λέξης;

Απάντηση A: KLIDARITHMOS
Απάντηση B: KLEIDARITHMOS
Απάντηση C: KLIDARITIMOS
Απάντηση D: CLEIDARITHMOS

Έχοντας φτιάξει την κασέτα δεδομένων, χρησιμοποιήστε το Πρόγραμμα 6.2 για ένα αυτο-τεστ - χωρίς να χρησιμοποιήσετε λεξικό.

Παράρτημα A

Γλωσσάρι

άμεσα αρχεία (direct files): Τρόπος οργάνωσης των αρχείων που επιτρέπει να εντοπιστεί αμέσως οποιαδήποτε εγγραφή, ανεξάρτητα απ' τη θέση της στο αρχείο.

αναδιοργάνωση αρχείου (file reorganisation): Μέθοδος ή πρόγραμμα που τροποποιεί την όλη δομή του υπάρχοντος αρχείου.

αντικειμενικός κώδικας (object code): Η μετάφραση του πηγαίου κώδικα σε κώδικα μηχανής.

αριθμητικά δεδομένα (numerical data): Το υποσύνολο εκείνο των χαρακτήρων που περιέχει μόνο τα ψηφία από 0 έως και 9.

αρχείο (file): Συλλογή πληροφοριών σχετικών μεταξύ τους.

αρχείο δεδομένων (data file): Αρχείο που περιέχει δεδομένα προσιτά μόνο από ένα μόνιμο πρόγραμμα.

αρχείο με περιεχόμενα (indexed file): Αρχείο όπου οι εγγραφές γίνονται προσιτές αφού συμβουλευτεί ο χρήστης ένα ξεχωριστό αρχείο περιεχομένων. Οι εγγραφές δεν χρειάζεται να είναι σε προκαθορισμένη σειρά.

βάση δεδομένων (data base): Όρος ο οποίος στους κύκλους των σπιτικών υπολογιστών είναι ουσιαστικά συνώνυμος με τον όρο σύστημα αρχειοθέτησης.

γλώσσα assembly (assembly language): Μία πιο φιλική προς το χρήστη μέθοδος για πέρασμα σε κώδικα μηχανής.

δείκτης (pointer): Μεταβλητή που αντιπροσωπεύει τη διεύθυνση μιας άλλης μεταβλητής.

δευτερεύον πεδίο-κλειδί (secondary key field): Ένα πεδίο που χρησιμοποιείται μάλλον για ταξινόμηση και όχι για προσδιορισμό.

δημιουργία αρχείου (file creation): Η εισαγωγή προκαταρκτικών πληροφοριών, όπως επικεφαλίδες πεδίων κ.τλ.

διαδοχική αναζήτηση (sequential search): Αναζήτηση που αρχίζει απ' την πρώτη εγγραφή και συνεχίζει διαδοχικά ώσπου να βρεθεί η επιθυμητή εγγραφή.

δυαδική αναζήτηση (binary search): Μία γρήγορη μέθοδος αναζήτησης που απαιτεί ταξινομημένη λίστα.

εγγραφή (record): Υποδιαίρεση ενός αρχείου δεδομένων.

εξ ορισμού (default): Οι φυσικές συνθήκες που επικρατούν στο σύστημα όταν απουσιάζουν αντίθετες διαταγές.

επεξεργασία δεδομένων (data processing): Όρος που αρχικά χρησιμοποιούνταν για να καλύψει την αποθήκευση και επεξεργασία πληροφοριών σε διάκριση με την εκτέλεση πράξεων.

επικεφαλίδα πεδίου (field heading): Ο τίτλος που ορίζει τη σημασία ενός πεδίου δεδομένων.

ιεραρχικό (hierarchical): Ένα σύστημα που δομείται κατά σειρά σπουδιαότητας.

κώδικας μηχανής (machine code): Πρόγραμμα γραμμένο σε μία μορφή που αναγνωρίζεται αμέσως απ' τον μικροεπεξεργαστή.

μέγεθος αρχείου (file size): Κανονικά, ο αριθμός των εγγραφών που περιέχει το αρχείο.

μεταγλώσσα (meta language): Περιεκτικό σύνολο συμβόλων που χρησιμοποιούνται για να περιγράψουν τις λέξεις - κλειδιά της **BASIC** κατά τρόπο όχι διφορούμενο.

baud: Η ταχύτητα του ενός bit πληροφοριών ανά δευτερόλεπτο.

byte: Ομάδα των 8 bit.

μπλοκ (block): Μονάδα που αποτελείται από έναν ορισμένο αριθμό bytes. Π.χ. στον Amstrad ένα μπλοκ είναι 2K bytes.

πεδίο (field): Υποδιαίρεση της εγγραφής.

πεδίο-κλειδί (key field): Ένα πεδίο που προσδιορίζει μονοσήμαντα μία εγγραφή.

πηγαίος κώδικας (source code): Ένα πρόγραμμα γραμμένο σε assembly ή σε γλώσσα ψηλότερου επιπέδου.

Πλάτος πεδίου (field width): Ο αριθμός των χαρακτήρων που έχουν κατανεμηθεί σ' ένα πεδίο.

προσωρινή μνήμη ταινίας (tape buffer): Μία περιοχή της μνήμης στην οποία γίνεται μεταφορά πληροφοριών από και προς την ταινία

πτητική (volatile): Η μνήμη στην οποία τα δεδομένα χάνονται όταν διακοπεί η παροχή ρεύματος στον υπολογιστή.

σήμα αναμονής (prompt): Μήνυμα στην οθόνη που ζητάει κάποια είσοδο από το πληκτρολόγιο

σημαία (flag): Στοιχεία δεδομένων που μπορεί να ισοδυναμεί ή με ναι ή με όχι.

συλλογή αχρήστων (garbage collection): Περιοδική απαλοιφή των πλεοναζόντων αλφαριθμητικών και συνακόλουθη αναταξινόμηση εκείνων που απομένουν.

ταξινόμηση με ανταλλαγή (exchange sort): Ταξινόμηση βασισμένη σε συγκρίσεις μεταξύ γειτονευόντων όρων.

υποαλφαριθμητική (substring): Ομάδα χαρακτήρων που ανήκει σε μία αλφαριθμητική.

υποαρχείο (subfile): Αρχείο που περιέχει μονάχα μία επιλογή εγγραφών από ένα κύριο αρχείο.

Φιλικό προς το χρήστη (user friendly): Η έμφυτη ικανότητα ενός προγράμματος να αναγνωρίζει και να ανέχεται την ανθρώπινη αδυναμία.

ψευδομεταβλητή (pseudo variable): Μεταβλητή της οποίας μπορεί να αλλάξει η τιμή όχι όμως και ο χαρακτήρας.

ΠΑΡΑΡΤΗΜΑ B

ΚΩΔΙΚΟΙ ASCII

Δεκαδικός	Δεκαεξαδικός	χαρακτήρ.	Δεκαδικός	Δεκαεξαδικός	χαρακτήρ.	Δεκαδικός	Δεκαεξαδικός	χαρακτήρ.
32	20	Space	64	40	@	96	60	£
33	21	!	65	41	A	97	61	a
34	22	″	66	42	B	98	62	b
35	23	#	67	43	C	99	63	c
36	24	$	68	44	D	100	64	d
37	25	%	69	45	E	101	65	e
38	26	&	70	46	F	102	66	f
39	27	′	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
42	2A	*	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	−	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
47	2F	/	79	4F	O	111	6F	o
48	30	0	80	50	P	112	70	p
49	31	1	81	51	Q	113	71	q
50	32	2	82	52	R	114	72	r
51	33	3	83	53	S	115	73	s
52	34	4	84	54	T	116	74	t
53	35	5	85	55	U	117	75	u
54	36	6	86	56	V	118	76	v
55	37	7	87	57	W	119	77	w
56	38	8	88	58	X	120	78	x
57	39	9	89	59	Y	121	79	y
58	3A	:	90	5A	Z	122	7A	z
59	3B	;	91	5B	[	123	7B	{
60	3C	<	92	5C	\	124	7C	|
61	3D	=	93	5D	]	125	7D	}
62	3E	>	94	5E	∧	126	7E	~
63	3F	?	95	5F	_	127	7F	Delete

Παράρτημα C

Απαντήσεις στις ασκήσεις

1.1. Ήταν μη πτητική.
1.2. Η πρώτη εσωτερική και πτητική, η δεύτερη εξωτερική και όχι πτητική.
1.3. Το περιοδικό *BYTE* αρχικά χρησιμοποιούσε ένα συνηθισμένο κασετόφωνο για να αποθηκεύσει ψηφιακές πληροφορίες. Το περιοδικό τυπωνόταν στο Κάνσας Σίτυ.
1.4. 2K.
1.5. Δύο αρχείο κασέτας μπορούν να ανοιχτούν ταυτόχρονα
1.6. Έχει διαπιστωθεί τέλος αρχείου.
1.7. Η FRE (« ») οδηγεί πρώτα σε συλλογή αχρήστων, αντίθετα με την FRE (0).
1.8. Διότι οι κεφαλές εγγραφής δύο διαφορετικών μηχανημάτων μπορεί να μην είναι ακριβώς ευθυγραμμισμένες.
1.9. Λάθος.
2.1. Όχι.
2.2. Πρέπει να είναι μοναδικό.
2.3. Όταν υπάρχει μεγάλος αριθμός εγγραφών.
2.4. Η επεξεργασία είναι γρήγορη απ' τη στιγμή που το αρχείο φορτωθεί.
2.5. 102 χαρακτήρες.
2.6. Οριζόντια.
2.7. Όταν ο χρήστης δεν είναι βέβαιος για το τι συμβαίνει σ' ολόκληρο το πεδίο.
2.8. Γιατί η διαθέσιμη μνήμη είναι πιο περιορισμένη απ' ότι σε αρχεία καταχωρημένα σε περιφερειακές μνήμες.
3.1. Η γραμμή 300 πρέπει να αλλάξει σε DATA 13,2.
3.2. Να διαπιστώσει αν η εγγραφή είναι σε αρχείο.
3.3. Να απορρίψει μια κενή είσοδο, δηλ. όταν έχει πατηθεί ENTER πριν δοθούν τα δεδομένα.
3.4. Δεν μπορείτε να ταξινομήσετε ένα αρχείο με μία μόνο εγγραφή.
4.1. Να κατανείμει μόνιμα τις περιοχές προσωρινής μνήμης της κασέτας.
4.2. Να αποκαταστήσει την ταχύτητα των 1000 baud.
4.3. Για να μπορούν να δοθούν άλλες διαστάσεις στον πίνακα A$.
4.4. Για να εξασφαλίσει ότι η διορθωμένη γραμμή εμφανίζεται μέσα στο κέντρο των εστιγμένων γραμμών.
5.1. 20 εκατομμύρια.
5.2. 6 ώρες.
5.3. 500000.
5.4. Μπορεί να έχει μέτριες επιδόσεις σε σχεδόν διαταγμένους πίνακες.

5.5. Βελτιώνει την ανάμιξη των συνόλων για μεγάλες τιμές του Ν.
6.1. Δεν χρειάζεται επίσημη απάντηση.

ΚΥΚΛΟΦΟΡΟΥΝ ΑΠΟ ΤΟΝ ΔΕΚΕΜΒΡΙΟ 1985:

Το σύστημα δίσκων του Amstrad
του Ian Sinclair

Εισχωρεί στη γνώση του συστήματος των δίσκων, περιγράφει το λειτουργικό σύστημα CP/M, τρόπους αρχειοθέτησης από την BASIC και περιέχει διάφορα UTILITIES όπως πως να ξεκλειδώνετε προστατευμένα προγράμματα κ.λπ.

Γλώσσα μηχανής για αρχάριους στον Amstrad
του Steve Cramer

Απευθύνεται στον Αρχάριο που θέλει να μάθει την γλώσσα μηχανής του Amstrad. Περιγράφει τις αρχές προγραμματισμού σε γλώσσα μηχανής, τις οδηγίες του επεξεργαστού Z-80 κ.λπ.

Συστήματα αρχειοθέτησης και βάσεις δεδομένων για τον Amstrad.
των A. P. Stephenson και D. J. Stephenson
Αναλύει τον τρόπο σχεδίασης των προγραμμάτων αρχειοθέτησης και βάσεων δεδομένων, με πλήρη και λεπτομερή παραδείγματα.

Όλα ταβιβλία αναφέρονται στους υπολογιστές CPC 464, CPC 664, CPC 6128

ΚΥΚΛΟΦΟΡΟΥΝ ΑΠΟ ΤΟΝ ΙΑΝΟΥΑΡΙΟ 1986:

Το Εγχειρίδιο του CP/M **του Alan Miller**

Το Εγχειρίδιο του MS-DOS **του Allen King**

Μάθετε εύκολα τον IBM-PC **του Lean Scanlon**

Pascal για αρχάριους του Maik James.
Αυτό το βιβλίο απευθύνεται σ' όλους τους χρήστες της Pascal, τεχνικούς, επαγγελματίες, φοιτητές αλλά και χομπίστες. Κάθε κεφάλαιο αναλύεται με πολύ σαφήνεια και απλό τρόπο. Επίσης υπάρχουν παραδείγματα αλλά και ερωτήσεις που σας βοηθούν να γράψετε πολύ σύντομα τα δικά σας προγράμματα στην Pascal.

LOGO για αρχάριους του Boris Allan.
Απευθύνεται κυρίως στους νεαρούς χρήστες που θέλουν να γνωρίσουν αυτή τη γλώσσα, όπως είναι οι κάτοχοι του Amstrad CPC 664, 6128 όπως και του Atari 520 ST.

ΚΥΚΛΟΦΟΡΟΥΝ:

Γνωρίστε το ZX-microdrive σας
του Andrew Pennell
Έχει τόση σαφήνεια και τόσα πολλά παραδείγματα που το κάνουν εξ ίσου χρήσιμο τόσο για τον νεοφερμένο στη BASIC όσο και για τον έμπειρο προγραμματιστή
ΑΠΑΡΑΙΤΗΤΟ ΣΕ ΚΑΘΕ ΚΑΤΟΧΟ SPECTRUM

Δουλέψτε με τον Amstrad
Χρήσιμα προγράμματα και υπορουτίνες
του David Lawrence
Μια συλλογή από έξυπνα προγράμματα και υπορουτίνες που μπορούν να χρησιμοποιηθούν αυτούσιες στα προγράμματά σας όπως αποθήκευση δεδομένων, οικονομικά κ.λπ.

Μάθετε Basic με τον Amstrad
του Wynford James
Το βιβλίο απευθύνεται στον νέο κάτοχο του Amstrad και τον εισάγει στον προγραμματισμό μεθοδικά και σταδιακά, χωρίς να έχει τις αδυναμίες του manual του υπολογιστή.
Να τι έγραψε το αγγλικό περιοδικό Personal Computer World για το αγγλικό manual: «Οι αρχάριοι μάλλον θα χρειασθούν ένα άλλο βιβλίο να τους μάθει Basic».

Basic για αρχάριους
του A. P. Stephenson
Μεθοδικό και απλό απευθύνεται στον αρχάριο της Basic και τον οδηγεί στα μυστικά του προγραμματισμού.

Δουλέψτε με τον Commodore
Χρήσιμα προγράμματα και υπορουτίνες
του David Lawrence
Ένα βιβλίο με πολλά έτοιμα προγράμματα και υπορουτίνες που θα σας προσφέρουν μεγάλη βοήθεια στην κατασκευή των δικών σας προγραμμάτων. Περιλαμβάνει προγράμματα με αποθήκευση δεδομένων, οικονομικά, γραφήματα ακόμα και επεξεργασία κειμένου.

γραφικές τέχνες αφοι σπυρου
τηλ 97 51.207 - 97 01.593

A. P. STEPHENSON &
D. J. STEPHENSON

ΣΥΣΤΗΜΑ ΑΡΧΕΙΟΘΕΤΗΣΗΣ ΚΑΙ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΣΤΟΝ AMSTRAD

ΕΚΔΟΣΕΙΣ
ΚΛΕΙΔΑΡΙΘΜΟΣ